uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2018-2019

Trabajo Fin de Grado

"Seguimiento de vuelo complejo. Análisis de algoritmos avanzados para una aplicación en Quiroptología"

Jesús García Cuchillero

Tutor

Antonio Berlanga de Jesus

Campus de Colmenarejo, Septiembre de 2019

RESUMEN

El objetivo de este proyecto es analizar las distintas técnicas y algoritmos de seguimiento y detección de objetos en imágenes y vídeos pertenecientes al enfoque clásico de la visión artificial, con el fin de entender su funcionamiento y poder adaptarlas y aplicarlas a grabaciones nocturnas de vuelos complejos. Y de este modo determinar como de efectivas son estas técnicas para este dominio en particular.

Para el estudio y test se emplearan implementaciones basadas en el lenguaje de Python, junto con la biblioteca de código abierto OpenCV de visión artificial. Con la implementación se realizarán distintas pruebas para evaluar si el resultado es satisfactorio y extraer las conclusiones pertinentes para este trabajo.

Palabras clave

Detección, seguimiento, formas, rasgos, background, foreground, convolucionales, visión artificial, estados, centroide.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1 Motivación del trabajo

La idea de este trabajo surge en una conversación entre el tutor de este TFG y sus compañeros de departamento con unos conocidos que trabajan en control de plagas de la península.

Esta gente entre otros trabajos se ocupa de mantener un control del número de murciélagos que hay en la península. Para ello viajan a distintos lugares donde hay colonias de murciélagos y durante la noche sitúan una cámara nocturna enfocando la entrada de la colonia, y con unos sistemas de sonidos les hacen salir de la cueva poco a poco. En la cámara quedan grabados los murciélagos que salen de la cueva y que entran, y los trabajadores con un contador manual van pausando el video secuencia a secuencia contando los murciélagos y siguiéndolos por la pantalla tratando de no contar varias veces los mismos. Esta forma de hacerlo como se puede imaginar es muy lenta y tediosa.

Así con el enorme auge que están teniendo las técnicas de seguimiento y de detección en los últimos años, porque no intentar crear una aplicación que pudiese detectar a los murciélagos y contarlos de la misma forma que hacen estas personas, pero de forma automática.

Esta fue la idea inicial sobre la que se planteó este TFG.

1.2 Objetivos del trabajo

Los motivos expuestos en la sección anterior dejan claro el "porqué", es decir la necesidad que ha impulsado la realización de este trabajo, pero se necesita saber también el "que", que es exactamente lo que se necesita y cómo se puede satisfacer.

De la respuesta a esa pregunta es de donde se extraen los objetivos que marcarán las pautas de realización del proyecto.

En los últimos años las aplicaciones de detección han tenido un crecimiento y desarrollo inmensos. Han evolucionado enormemente tanto en precisión como en velocidad y eficiencia. Sin embargo estos nuevos sistemas son pesados de crear y muy difíciles de calibrar. No solo requieren de equipos electrónicos potentes para funcionar, si no que si se quieren hacer funcionar correctamente se necesitan enormes cantidades de datos para poder entrenarlos. En casos de objetos cotidianos, como personas, coches, perros… hay enormes bancos de datos disponibles para todo el mundo, e incluso hay modelos ya entrenados en estos mismos bancos. Pero el problema que se plantea en este documento no es común, los videos que se tratan tienen características únicas y por tanto se debería crear un modelo hecho a la medida del problema.

Es por eso que primero se propuso ver que tal funcionan técnicas más antiguas pero mucho más ligeras y variadas. Si su funcionamiento fuera prometedor, se podrían emplear

estos mismos sistemas para recopilar datos en proyectos futuros con sistemas más modernos. De esta forma esos trabajos estarían planteados sobre unas bases sólidas y tendrían una forma fácil de extraer información de esos datos que necesitan para poder funcionar debidamente. Así pues los objetivos de este proyecto quedaron marcados como:

1. Investigar las distintas técnicas y métodos de análisis de videos y detección de objetos existentes en la etapa clásica. Cuando se habla de investigar, no se hace referencia únicamente al entendimiento de su funcionamiento e implementación, es imperativo poder determinar teóricamente si el algoritmo o método es viable o no para intentar resolver el problema que se plantea.

   Para lograr esto se deben analizar las distintas operaciones y funciones que aplica cada uno de ellos y adaptarlos de forma que su funcionamiento se adecúe a las características del problema que se trata.

2. Conseguir crear al menos uno o varios sistemas que emplee estas técnicas de seguimiento y detección de objetos en video y comprobar si es capaz de identificar y seguir el vuelo de murciélagos en grabaciones de cámaras nocturnas (tonos grises) con un buen rendimiento, tanto en velocidad como en porcentaje de acierto.

   En este objetivo es importante distinguir las dos partes que deben componer el programa para que este pueda satisfacer este objetivo.

   - La detección o identificación, es la parte del programa que deberá ser capaz de, dada una instancia del vídeo, determinar cuántos de los objetos deseados hay en esa instancia y también en qué posición de la misma se encuentran.
   - El seguimiento o tracking, es la parte que dadas las posiciones de cada objeto por el algoritmo de detección debe ser capaz de marcar cada uno individualmente y seguirlo en las sucesivas instancias del video, siendo robusto a perdidas, por su parte, por parte del algoritmo de detección o por la salida de los objetos del video.

   Esto casi se puede podría establecer como dos objetivos diferentes, pero puesto que por sí solos no cumplen ningún objetivo real, es el ensamblado de ambas partes la que generará la solución óptima.

3. Lograr que el programa lleve la cuenta de la cantidad de objetos (murciélagos) que pasan por los vídeos. Este es el objetivo final y de los más importantes, el código resultante debe llevar la cuenta individual de los objetos los más acertadamente posible, es decir, ser capaz de distinguir de entre los objetos detectados lo suficiente como para no contarlos más de una vez, mientras estos

permanezcan el en video, aumentando el conteo global cada vez que aparece un nuevo individuo. Una vez más este debe ser robusto ante perdidas.

1.3 Estructura del documento

Esta memoria está dividida en 9 puntos, incluyendo esta sección de introducción, la memoria contendrá:

- **Contexto y estado de la cuestión:** Aquí se incluye una detallada descripción de las distintas tecnologías y técnicas que han surgido en este campo de la informática, las diferencias entre los distintos acercamientos al mismo objetivo y en cómo han evolucionado cada una de sus ramas. Se prestará especial atención a aquellas técnicas que se vayan a emplear como parte de las soluciones explicando en detalle su funcionamiento teórico.

- **Análisis del problema:** Aquí se explicaran en detalle el problema que se aborda en este proyecto, como son los videos que se deben analizar cuáles son las distintas formas que se puede abordar este tipo de análisis en función de las características de los mismos.

- **Diseño de la solución:** En esta sección se explicaran las decisiones que nos han llevado a escoger el diseño de las soluciones que mejor abordan este problema.
  Se justificará el por qué estos se han considerado las mejores posibles soluciones, siempre teniendo en cuenta los objetivos fijados en el apartado correspondiente a los mismos. Puesto que hay varias soluciones viables se mostrarán todas por igual, y más adelante se evaluarán los resultados y desempeño individual de cada una de ellas.
  Además en este apartado se explicara detalladamente la implementación de las mismas, incluyendo los elementos que las componen tales como las herramientas empleadas, librerías, funciones, frameworks, etc …

- **Pruebas:** En esta sección se realizarán distintas pruebas con el fin de evaluar y medir el funcionamiento de las soluciones implementadas, comprobando la calidad de los resultados y la cantidad de acierto que estos posean

- **Planificación y presupuesto:** En esta sección se mostrará la planificación que se ha seguido para la realización de este proyecto, y se incluirán las estimaciones, retrasos, solapamientos etc…
  Además se comentarán y calcularán los gastos estimados en la realización de dicho trabajo, personal, material, etc…

- **Entorno socioeconómico:** Se discutirán las posibles implicaciones de este proyecto a nivel económico si se considera que deba tener alguno. En caso

contrario se explicará el poruque no influye en ningún aspecto cocial, económoco o ambiental.

- **Conclusiones y trabajos futuros:** Se finalizara este documento con un análisis de los objetivos cumplidos desde varios puntos de vista, y también con una visión de posibles mejoras sobre la solución obtenida de cara al futuro. Y de trabajos que se servirán del mismo y lo emplearán como base.

- **Bibliografía:** Mención de todas las referencias bibliográficas empleadas de alguna manera en la realización de este proyecto siguiendo el estándar numérico marcado por el IEEE.

1.4 Marco regulador

En la realización de este trabajo se tomará en cuenta el nivel de legislación y normativa.

La razón de este proyecto es meramente de carácter académico todo en él está orientado a la investigación. En el supuesto de que alguno de los sistemas propuestos en el proyecto se emplease para la comercialización se debe recalcar que puesto que no se manejan ningún tipo de información de carácter personal o privado de ningún tipo, los datos que se manejan son de libre conocimiento y entregados sin ningún tipo de restricción de privacidad y el material empleado para ello está libre de patentes o restricciones, no habría que tener en cuenta ninguna legislación de protección de datos.

2. ESTADO DE LA CUESTIÓN

2.1 Contexto histórico

Hasta el momento se ha mencionado varias veces los conceptos de detección de objeto y seguimiento de objetos, queriendo decir que aunque ambas están relacionadas, son fundamentalmente distintas. Entonces qué es lo que las relaciona, porque ambas son necesarias para este proyecto.

Bien, lo primero que se debe saber es que esos dos conceptos y otros más que se emplearan a lo largo de este documento, están englobados en un campo de la tecnología de computadores mucho mayor, la Computer Vision o visión computacional (CV). Todo con lo que se trabajará en este proyecto son partes que pertenecen este campo de la ciencia de computadores [1].

La visión computacional es un campo interdisciplinario, es decir que está relacionado con otros campos. Es un campo que grosso modo trata de lograr que las máquinas obtengan un alto nivel de comprensión partir de imágenes digitales o videos.

Las tareas principales de la visión artificial [2] incluyen métodos para adquirir, procesar, analizar y comprender imágenes digitales y extraer datos de alta dimensión del mundo real para producir información numérica o simbólica, para por ejemplo realizar una toma de decisiones. Hay que entender que en este contexto, esto se traduce en la transformación de imágenes visuales, en una descripción del mundo que puede ser interpretada por otros procesos y así realizar la acción adecuada, en definitiva la comprensión de lo que se está viendo. Para los ordenadores esto es descomponer la información contenida en una imagen en datos que puedan emplear modelos construidos para tal fin, mediante la geometría, estadística, física y teoría del aprendizaje.

Desde la perspectiva de la ingeniería, este campo busca automatizar las tareas que puede realizar el sistema visual humano. La visión artificial se ocupa de la extracción automática, el análisis y la comprensión de información útil de una sola imagen o una secuencia de imágenes. Implica el desarrollo de una base teórica y algorítmica para lograr la comprensión visual automática.

Como disciplina científica, la visión computacional se ocupa de la teoría detrás de los sistemas artificiales que extraen información de las imágenes. Los datos de imagen pueden tomar muchas formas, como secuencias de video, vistas desde múltiples cámaras o datos multidimensionales de un escáner médico. Como disciplina tecnológica, la visión artificial busca aplicar sus teorías y modelos para la construcción de sistemas de visión artificial.

Algunos subdominios de la visión por computador incluyen reconstrucción de escenas, detección de eventos, seguimiento de video, reconocimiento de objetos, estimación de pose 3D, aprendizaje, indexación, estimación de movimiento y restauración de imágenes. Varios de los cuáles serán los que se traten en este presente proyecto.

La visión artificial podría decirse que nació a finales de la década de 1950, comenzó en universidades que eran pioneras en inteligencia artificial. Estaban enfocadas a imitar el sistema visual humano, como un trampolín para dotar a los robots de un comportamiento inteligente.

En 1959, Russell Kirsch y sus compañeros crearon un aparato que permitía transformar una imagen en un conjunto de cuadrículas o grids de números, un sistema de lenguaje que las máquinas podían entender, es gracias a ellos que ahora se pueden procesar imágenes de múltiples formas, fueron los primeros en escanear una imagen con éxito, la foto de su hijo, figura 1.

Figura 1: Russell Kirsch hijo

En 1963 Lawrence Roberts describió un proceso para recrear una imagen en 3D a partir de fotografías en 2D, básicamente reducía el mundo visual a una simple representación de formas geométricas. Su objetivo era procesar las imágenes en 2D como dibujos de líneas y después crear representaciones en 3D de estas líneas. El defendía que este proceso era un buen punto de partida para investigaciones futuras, lo cual acabó siendo cierto.

En los sesenta cuando la AI (Inteligencia Artificial) se convirtió en una disciplina académica, los investigadores se volvieron muy optimistas sobre el futuro del campo, se creía que no se tardaría mas de 25 años en conseguir que un ordenador fuese tan inteligente como un humano. De hecho en 1966 Seymour Parpert un profesor del laboratorio del MIT, decidió lanzar un programa de verano para resolver el problema de la visión computacional en unos meses, creía que esto podría lograrse al conectar una cámara a una computadora y hacer que esta describiese lo que veía. Junto con un grupo de estudiantes crearon una plataforma que automáticamente realizase segmentación de fondo y frente (backgroun/foreground) y extrajese objetos no superpuestos de imágenes del mundo real. El proyecto no fue un éxito sin embargo muchos concuerdan en que fue el nacimiento de la visión artificial como campo científico.

Lo que distinguió la visión artificial como campo predominante en el mundo del procesamiento de imágenes digitales en ese momento fue el deseo de extraer una estructura tridimensional de las imágenes con el objetivo de lograr la comprensión completa de la escena. Los estudios en la década de 1970 formaron los cimientos iniciales de muchos de los algoritmos de visión artificial que existen hoy en día, como la extracción de bordes de imágenes, el etiquetado de líneas, el modelado no poliédrico y poliédrico, la representación de objetos como interconexiones de estructuras más pequeñas, flujo óptico y detección del movimiento.

La siguiente década vio sus estudios basados principalmente en un análisis matemático más rigurosos y aspectos cuantitativos de la visión artificial. Estos incluyen el concepto de espacio de escala, la inferencia de formas a partir de varias señales tales como sombreado, textura y enfoque, y modelos de contorno conocidos como serpientes.

En la década de 1990, algunos de los temas de investigación anteriores se volvieron más activos que los otros. La investigación en reconstrucciones proyectivas en 3D llevó a una mejor comprensión de la calibración de la cámara. Con el advenimiento de los métodos de optimización para la calibración de cámaras. Esto condujo a métodos para reconstrucciones en 3D de escenas a partir de múltiples imágenes. Se avanzó en el problema de la correspondencia estéreo densa y otras técnicas estéreo de múltiples vistas. Al mismo tiempo, se utilizaron variaciones de corte de gráfico para resolver la segmentación de la imagen. Esta década también marcó la primera vez que se usaron técnicas de aprendizaje estadístico en la práctica para reconocer caras en imágenes.

Hacia finales de la década de 1990, se produjo un cambio significativo con más interacción entre los campos de los gráficos por ordenador y la visión artificial. Esto incluía la representación basada en imágenes, la transformación de imágenes, la interpolación de vistas, la costura panorámica de imágenes y la representación temprana de campos de luz.

Alrededor de 1999 muchos investigadores dejaron de intentar reconstruir objetos creando modelos 3D de ellos y en cambio dirigieron sus esfuerzos hacia el reconocimiento de objetos basado en formas (feature-based). El trabajo de David Lowe fue especialmente importante en esto, más concretamente el documento que publicó titulado “Object recognition from Local Scale-Invariant Features” cuya traducción sería más o menos “Reconocimiento de objetos a partir de formas locales invariables”, describía un sistema de reconocimiento visual que usaba formas locales que no variasen por la rotación, localización, partición o cambios en la iluminación. Estas formas de acuerdo a Lowe son algo similares a las propiedades de las neuronas en la parte inferior del cortex temporal, que está relacionado con el proceso de detección de objetos en la visión primaria.

Así se llegó a la primera década del segundo milenio, es en este tramo donde la visión artificial avanza de forma palpable, y aparecen los primeros modelos sólidos y funcionales de **detección y seguimiento**. Es aquí donde surgen la mayoría de los algoritmos que pertenecen a la ahora conocida como aproximación tradicional de la visión artificial.

2.2 Algoritmo de detección

Para contextualizar los algoritmos de detección que ese explorarán en este documento, se hará referencia al estudio de "Object detection in 20 years: a Survey" [3] donde se explica muy bien el estado actual de esta materia.

La detección de objetos es una tarea muy importante dentro de la visión computacional, que afronta el problema de detectar instancias de objetos visuales de cierta clase, tales como personas, animales o coches, en imágenes digitales. El objetivo es desarrollar modelos computacionales y técnicas que provean de una de las informaciones más básicas para las aplicaciones de visión computacional, que son los objetos y donde están.

Como uno de los problemas fundamentales de la CV, la detección es esencial para otras tareas como son: la segmentación de instancias, captura de imágenes, el seguimiento de objetos, etc… Desde el punto de vista de las aplicaciones, la detección de objetos pueden agruparse en dos tópicos de investigación, la detección de objetos general y las aplicaciones de detección, donde la primera aspira a explorar métodos de detectar diferentes tipos de objetos bajo un framework único para simular la visión humana y la cognición, y la segunda refiere a escenarios de aplicación específicos como reconocimiento facial, reconocimiento de texto, de peatones, etc..

En los últimos años el rápido crecimiento de las técnicas de aprendizaje profundo (Deep learning) han traído sangre nueva por así decirlo, a la detección de objetos, dándoles un increíble empuje y convirtiéndose en un foco de investigación sin precedentes.

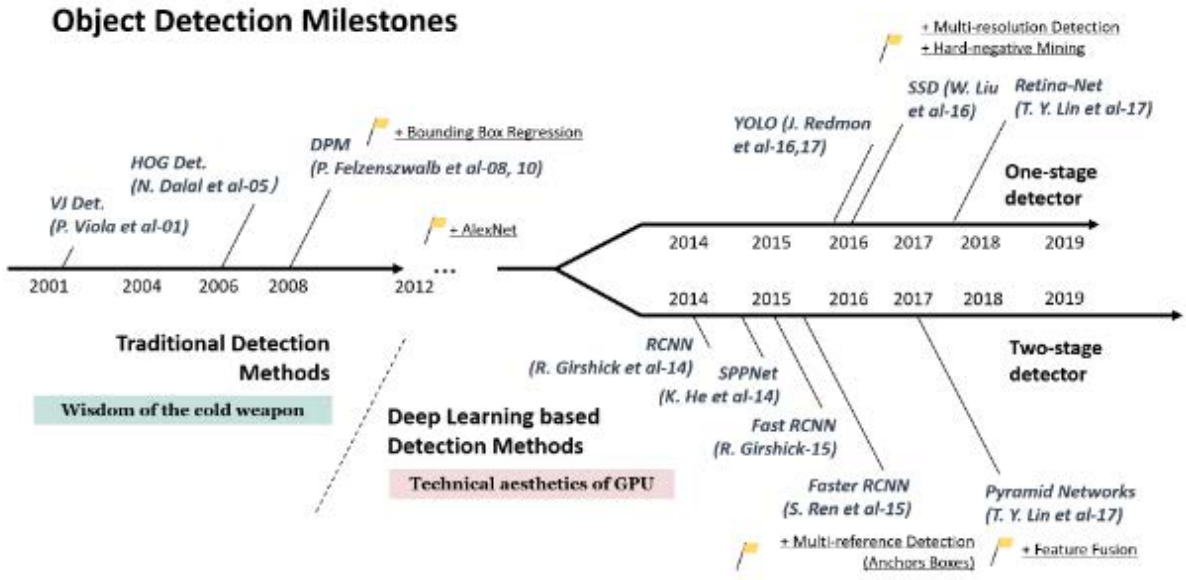


Figura 2: Línea temporal OD

En las últimas dos décadas, está extendida la aceptación de que el progreso de la detección de objetos se ha dividido en dos periodos históricos, el periodo de **la detección de objetos tradicional** (antes del 2014), y el periodo **de la detección basada en aprendizaje profundo** (después del 2014). Tal como se ilustra en la figura 2.

Detección de objetos tradicional

Si se piensa en la detección de objetos de hace 20 años, la mayoría de ellos estaban construidos sobre sistemas basados en características manuales. Debido a la falta de representación de imágenes efectiva, no había más opción que diseñar formas sofisticadas de representación, y una serie de técnicas para acelerar el proceso llevando al límite el uso de los recursos computacionales.

Algunos de los algoritmos más relevantes de este periodo son:

- Detector de Viola Jones

En 2001 P.Viola y M. Jones consiguieron crear con éxito un detector en tiempo real para detectar rostros humanos por primera vez, sin restricciones de color o raza. Este detector era decenas de veces más rápido que el resto de algoritmos de su tiempo y mucho más preciso.

Este algoritmo seguía el camino más directo posible, deslizándose por todas las posibles localizaciones de la ventana para ver la esta poseía un rostro. Parece un proceso sencillo, pero el cálculo de detrás estaba lejos del poder de computación de su tiempo.

El algoritmo funcionaba empleando tres características principales, la imagen integral que se emplea para acelerar el proceso de filtrado de cajas, o el proceso de convolución. La selección de rasgos, se empleaba el algoritmo de Adaboost para seleccionar las formas que más ayudan a representar el objeto que se busca. Y la detección en cascada, un paradigma para reducir el coste computacional.

- Detector de HOG

Histograma de gradientes orientados, fue propuesto en 2005 por N.Dalan y B.Triggs. HOG puede considerarse como una mejora de la transformación de rasgos de escala invariante y contextos de formas de su tiempo. El descriptor de HOG está diseñado para procesar una rejilla o grid densa de células uniformemente espaciadas y usar normalización de contraste local superpuesto para mejorar la precisión. Aunque puede usarse para detectar una gran cantidad de objetos, fue inicialmente diseñado para la detección de peatones. Ha sido un importante pilar para muchos detectores de objetos por muchos años.

- DPM (Deformable Part-based Model)

Fue propuesto originalmente por P.Felzenswalb en 2008 como una extensión de HOG y más adelante se le implementaron una gran cantidad de mejoras.

Este algoritmo sigue el principio de “divide y vencerás”, donde el entrenamiento puede ser simplemente considerado como el aprendizaje de la forma de descomponer un objeto y la inferencia como el proceso de detectar las partes que juntas componen el objeto. Por ejemplo detectar un coche sería detectar las ventanas, ruedas y cuerpo.

La forma típica del DPM consiste en un filtro raíz y un número de filtros de partes. En lugar de especificar manualmente la configuración de los filtros de partes, se desarrolla

un débil entrenamiento supervisado donde las configuraciones de los filtros de partes pueden aprenderse automáticamente como variables latentes R. Para acelerar la detección, R.Girshick desarrollo una técnica para "compilar" los modelos de detección en uno mucho más veloz que implementaba una arquitectura en cascada, que lo aceleraba hasta diez veces más, sin perder precisión.

Aunque los detectores de hoy en día han sobrepasado DPM en términos de precisión, muchos de ellos siguen profundamente influenciados por su valor y mezcla de modelos.

En este periodo hay muchos más algoritmos y enfoques además de estos tres, hay que pensar que algunos modelos se ajustan mejor a unas tareas que otros. Algunos de los cuales se tratarán en más profundidad en este documento.

Segunda parte. Modelos basados en CNN

Entre los años 2010 y 2012, el progreso de los sistemas de detección empezó a ralentizarse. Hasta que en 2012 hubo un resurgimiento de las redes neurales convolucionales o CNN. Estas redes neurales pueden aprender de forma robusta, incluso representaciones de formas de alto nivel de una imagen. Una pregunta que surgió, fue si esto podía aplicarse a la detección de objetos. En 2014 se propuso un método de regiones con CNN para la detección de objetos, y desde entonces la detección de objetos ha tenido un avance sin precedentes.

En la era del Deep learning, la detección de objetos puede dividirse en dos grupos, aquellos algoritmos que funcionan en dos tiempos o estados, y aquellos que lo hacen en solo uno.

CNN basados en detección en dos tiempos.

RCNN: La idea detrás de esta propuesta es simple. Se empieza con una extracción de un conjunto de objetos propuestos para una búsqueda selectiva. Cada propuesta es re escalada para adecuarse al modelo y se le pasan al modelo de CNN entrenado con ImageNet para extraer los rasgos. Y finalmente se le pasa a un clasificador lineal SVM para predecir la presencia de un objeto en cada región y reconocer la categoría de los mismos. Este proceso tiene como principal problema el tiempo, es muy lento.

Por eso en los años siguientes se propusieras dos variaciones, Fast RCNN y otra aún más veloz Faster RCNN. Que siguen el mismo principio pero permitiendo realizar tareas al mismo tiempo y varían los conjuntos de datos que se emplean.

SPPNet: Apareció en 2014, y su mayor contribución al campo fue la introducción de la capa de la pirámide espacial de agrupamiento (SPP) que permite a las CNN generar representaciones de longitud arreglada sin necesidad de tener que re escalarlas. De esta forma cuando se usa SPPNet, se evita tener que procesar repetidamente las convoluciones de los rasgos.

Feature Pyramid Networks: Fue propuesto en 2017, con el fin de crear un modelo que no solo usase la detección en la capa de más arriba. Con este fin se presentó una arquitectura de arriba abajo con conexiones laterales para construir semánticas de alto nivel en todas las escalas.

CNN basados en detección en un tiempo.

You Only Look Once (YOLO): Yolo fue propuesto en 2015 por R.Joseph. Fue el primer detector en un solo paso, y era muy rápido. Abandonaba el paradigma de los modelos anteriores de detección y verificación. En su lugar seguía un principio muy distinto, aplicar una sola red neural a la totalidad de la imagen. Esta red dividía la imagen en regiones y generaba cajas delimitadoras de predicción y probabilidades para cada región simultáneamente. Más adelante se propusieron otras dos versiones de mejora. A pesar de su velocidad YOLO sufre de una bajada en su precisión especialmente en objetos pequeños.

SSD (Single Shot multibox Detector): La principal contribución de este modelo es la introducción de técnicas de detección de multi referencias y multi resolución, que mejoran significativamente la precisión de la detección, especialmente para objetos pequeños. La principal diferencia entre este modelo y sus predecesores es que detecta los objetos de diferentes escalas en distintas capas de la red, mientras que los anteriores solo realizan la detección en las capas más altas.

RetinaNet: A pesar de su gran velocidad, los detectores en un estado sufrieron mucho su precisión frente a los de dos estados. En esta propuesta del 2017 se descubrió por qué esto es así. En esta se defendía que el extremo desbalanceo entre las clases de foreground (parte frontal) y background (fondo) durante el entrenamiento de los detectores era la causa central. Para resolver esto se creó una función de perdida llamada perdida focal, que servía para que el detector pusiera más atención en los ejemplos difíciles y desclasificados durante el entrenamiento, lo que permitió que los algoritmos de un solo tiempo lograsen una precisión equivalente a los de dos tiempos.

Y así se llega a 2019. Muchos de estos algoritmos se han empleado para crear sistemas de detección enormemente eficaces y seguro que seguirán usándose durante varios años. Sin embargo con la velocidad con la que se está desarrollando este campo de la ciencia, seguro que surgen nuevos enfoques que los superarán en muy poco tiempo. Nada en este campo permanece estable mucho tiempo.

Como ya se mencionó al principio del documento, los intereses de este trabajo residen en corroborar que se pueda detectar y realizar un conteo lo más correcto posible de los murciélagos en los videos proporcionado empleando distintas técnicas de la era tradicional de la detección de objetos, que en principio son computacionalmente más ligeros, para en futuros proyectos emplearlos para recopilar ejemplos para algoritmos basados en CNN.

2.3 Algoritmos de Tracking (Seguimiento)

El seguimiento de objetos en videos es una tarea bastante destacada dentro del campo de la visión artificial. Como una tecnología interdisciplinar, combina elementos del procesamiento de imágenes, reconocimiento de patrones, inteligencia artificial, control automático y otras áreas del conocimiento. El seguimiento de objetos también ha tenido bastante aplicación práctica en muchos campos, vigilancia de video, interacción humano-máquina, trafico inteligente, visión de navegación robótica, precisión de armas guiadas, etc… La investigación en el campo de los algoritmos de seguimiento tiene gran importancia teórica y significado práctico.

El seguimiento de objetos en video precisa de los pasos de detección, extracción y reconocimiento para obtener información de movimiento precisa, parámetros como son la velocidad, posición, etc… y llevar a cabo un análisis del proceso correspondiente en caso de que se desee llegar a implementaciones que entiendan el comportamiento de objetos deseados.

El seguimiento también es un proceso difícil y puede llegar a ser muy complicado por si se dan formas de objetos complejas, movimientos irregulares, cambios de iluminación, oclusión de objetos, etc.

En los últimos años con el alza en la investigación de las tecnologías de imagen digital y la visión artificial, el campo del seguimiento ha tenido un desarrollo bastante pronunciado y se han propuesto muchos algoritmos de probada calidad. De hecho hay tantos que hay múltiples formas de clasificarlos y ordenarlos. Algunos permiten el seguimiento de múltiples objetos, otros además permiten marcarlos como únicos, otros son mas potentes pero solo funcionan para seguir un único objeto, otros son capaces de guardar registros de posiciones anteriores para predecir trayectorias, etc…. Como ya se ha declarado en secciones anteriores, en este documento se mirarán aquellos métodos que fueron creados para combinarse con los métodos de detección tradicionales, descartando así aquellos que requieren de deeplearning para funcionar. También se limitará a aquellos algoritmos que puedan ser implementados con la herramienta de OpenCV y Python que es con la que se trabaja.

Aclarado esto, se va a seguir la clasificación que se realizó en el estudio *"Systematic Survey on object tracking Methods in video"* [4] que es un sistema de clasificación que se ha usado en muchos otros estudios.

El seguimiento puede ser definido como el problema para aproximar el camino de un objeto en el plano de la imagen mientras se mueve por la escena. El objetivo en la mayoría de los casos es generar una ruta para el objeto en concreto, encontrando su posición en cada fotograma del video. El seguimiento de objetos puede clasificarse en tres tipos, **seguimiento de punto, seguimiento basado en kernel y seguimiento basado en silueta**. Como se ilustra en la figura 3. Es importante remarcar que el seguimiento de puntos necesita que el detector funcione en todos los frames del video, mientras que los otros dos solo necesitan que el detector funcione cuando un objeto entra en la escena por primera vez. Esto será determinante cuando se vaya a implementar un tracker para las pruebas y se explicará el por qué cuando se hable del problema que se trabaja en concreto.

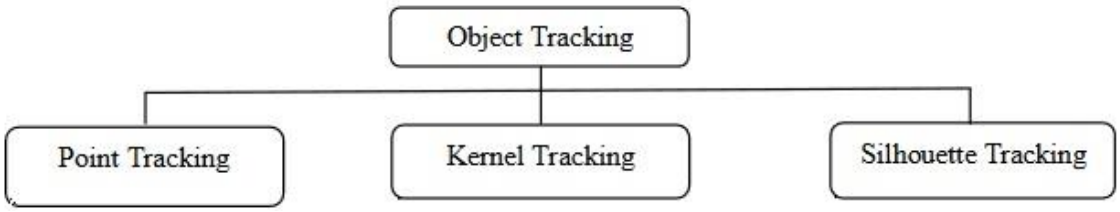


Figura 3: Clasificación de Trackers

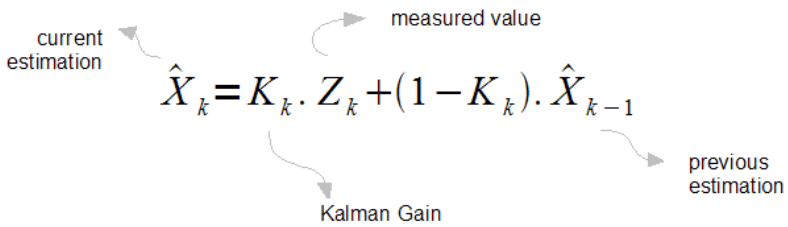
2.3.1 Seguimiento de puntos

Como su nombre indica esta forma de seguimiento se centra en seguir a los objetos fijándose ya sea en los puntos de la figura (rasgos) o en el centro de la misma, estas formas de seguimiento tienen la ventaja de ser muy ligeros y permiten llevar un registro de posiciones de los puntos. Sin embargo sufren ante la oclusión de objetos y las falsas detecciones por parte del detector.

Filtro de Kalman: [5] En matemáticas el filtro de Kalman es un algoritmo que se emplea para actualizar, observación a observación, la proyección lineal de un sistema de variables sobre un conjunto de información disponible, según se va obteniendo nueva información. Para ello es necesario representar el modelo en la formulación conocida como espacio de los estados. El filtro de Kalman permite calcular de un modo sencillo la verosimilitud de un modelo dinámico lineal, de una ecuación o varias, lo que permite estimar los parámetros de dicho modelo, así como obtener predicciones para el mismo.

Aplicado al seguimiento de objetos, el filtro de Kalman permite modelar un seguimiento basándose en la posición y velocidad de un objeto y predecir donde es más probable que este. Modela las posiciones y velocidad en el futuro mediante el empleo de gaussianos. Cuando recibe una nueva lectura puede usar probabilidades para asignar medidas a sus predicciones y actualizarse a sí mismo.

El filtro de Kalman posee unas matemáticas bastante complejas, es suficiente saber que a partir del vector de posición y velocidad, se llega a obtener esta ecuación:



El filtro de Kalman se compone de dos pasos, el de predicción y el de corrección.

En el primer paso el estado se predice con un modelo dinámico, el paso de predicción emplea el modelo del estado para predecir el nuevo estado de las variables.

De forma similar el paso de corrección usa la observación actual Z para actualizar el estado del objeto

Los componentes básicos de un Filtro de Kalman son, el vector de estados, el modelo dinámico y el modelo de observación como se ilustra en la figura 4:

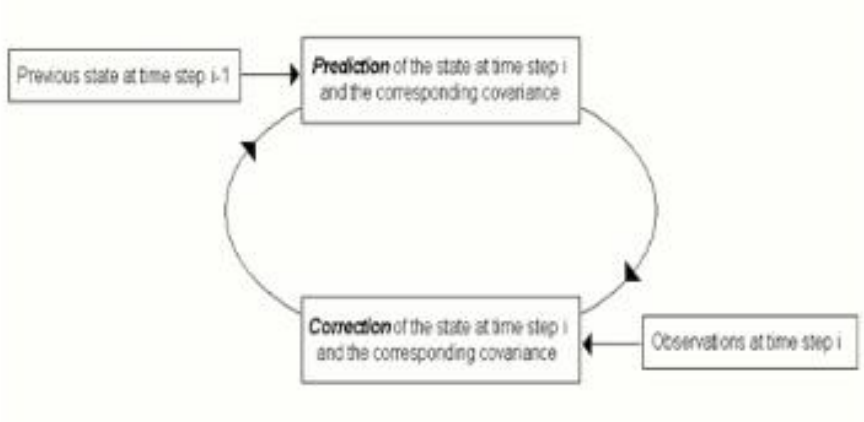


Figura 4: Ciclo de Kalman

2.3.2 Seguimiento de Kernel

Este tipo de algoritmos se emplean para seguir un objeto en movimiento que está representado por una región embrionica, de un fotograma al siguiente. El tipo de movimiento que realiza el objeto es de tipo paramétrica, como la traslación, no funciona bien con la rotación y similares.

Estos algoritmos se diferencian por el tipo de representación que usan, el número de objetos que siguen y el método usado para aproximar el movimiento de los objetos.

Simple coincidencia de modelo (template): Si también se puede emplear para seguir una imagen fija entre frames. Es un método de fuerza bruta que examina una región de interés del video y sirve para encontrar pequeñas partes de una imagen que coinciden con el modelo o template empleado.

Meanshift y Camshift: Estos dos algoritmos son casi iguales. Meanshift trata de encontrar un área en un video que sea similar a un modelo previamente inicializado, la región de la imagen a seguir se representa con un histograma. Se emplea un procedimiento de cuadrículas ascendente que trata de maximizar la valoración de similitud entre el modelo y la región de la imagen que se esté observando. Camshift difiere en que varía el tamaño de la ventana si el tamaño del objeto cambia, cosa que meashift no hace.

Suport Vector Machine: Es un método de clasificación que da una serie de conjuntos de entrenamientos positivos y negativos. Maneja bien el seguimiento individual, y la oclusión parcial, pero necesita ser inicializado y entrenado.

Layering Based Tracking: Este es otro método del kernel para mutiobjetivos. Cada capa esta formada de una representación de formas (elipse), movimiento como la traslación y la rotación, y capas de apariencia basadas en la intensidad. Es capaz de manejar oclusión total y el seguimiento múltiple.

2.3.1 Seguimiento basado en siluetas

Algunos objetos tienen formas complejas como las manos, dedos, hombros, que no pueden ser bien definidos mediante simples formas geométricas

Seguimiento de contorno:

Es un método que iterativamente pasa un contorno primario de un frame previo a su posición en el frame actual. Este proceso del contorno requiere que cierta cantidad del objeto en el frame actual se superponga con la región del frame previo. La mayor ventaja de este modelo es su flexibilidad para manejar las siluetas de distintos objetos.

Coincidencia de forma:

Este acercamiento examina el modelo del objeto en el frame actual. El funcionamiento de la coincidencia de la forma es muy similar al de coincidencia de modelos en el acercamiento de kernel. La detección de silueta esta llevada a cabo por substracción del fondo, los modelos de los objetos están basados en las formas derivadas de las funciones de densidad, y ejes de las máscaras.

2.3 Aplicaciones prácticas y proyectos similares

La única y más importante aplicación de la visión artificial es la comprensión de imágenes o videos que no dejan de ser una consecución de imágenes. Entender una imagen es algo muy complejo como ya se ha explicado, los estudios en esta materia se han dividido en ciertas tareas imprescindibles para la comprensión de imágenes. Hay múltiples tareas para esto, habitualmente se dividen en tareas de bajo nivel que suelen emplearse en las otras tareas, las de alto nivel.

Algunas tareas de bajo nivel son: limpieza de imágenes, segmentación de imágenes, análisis de histogramas, transformación de imágenes, detección de bordes y contornos, etc…

Las tareas de alto nivel a menudo se sirven de las de bajo nivel para funcionar, algunas son: Detección de objetos, reconocimiento de objetos, segmentación de objetos y localización, seguimiento de objetos, extracción de rasgos, corrección de formas y color, reconstrucción de formas, etc…

Las aplicaciones de la visón artificial suelen estar basadas en estas tareas de alto nivel. En los últimos años debido a los enormes avances en el campo del Deep learning, la mayoría de estas emplean este tipo de técnicas para mejorar su aplicación. Las técnicas más antiguas han quedado relegadas a ser usadas para pe procesar los datos e imágenes con los que se alimentarán las redes de Deep learning.

La detección de objetos que es lo que más se maneja en este proyecto tiene una gran variedad de usos en la industria, como por ejemplo [6]:

Reconocimiento óptico de caracteres: A menudo abreviado como OCR, es la conversión mecánica o electrónica de imágenes de tipado, escritura manual o impresiones, a texto máquina, tanto de un documento escaneado, una foto del mismo, una escena, o incluso de un video.

Se usa sobre todo para generar información de papeles impresos, como pasaportes, facturas, estados de cuenta bancarios, o cualquier documentación que a menudo se imprime en textos.

Figura 5: Caracteres

Coches autónomos: Uno de los mejores ejemplos que porque se necesita la detección de objetos es para la conducción autónoma. Para conseguir que el coche decida qué es lo que debe hacer, acelerar frenar o girar, necesita conocer la posición de todos los objetos a su alrededor y que son esos objetos. Para este tipo de detección sería esencial entrenar al sistema a reconocer objetos como coches, peatones, luces de tráfico, signos de la carretera, ciclistas, etc..

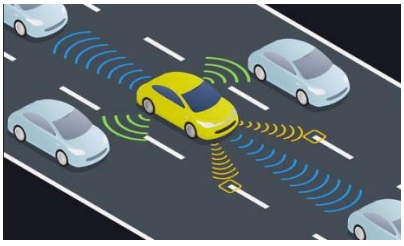


Figura 6: Coche autónomo

Imágenes médicas: Las herramientas de procesado de imágenes médicas, están creciendo en uso y adquiriendo importancia en asistir tareas clínicas como el diagnóstico, planificación de terapia e intervenciones de imagen guiada. Precisos, robustos y rápidos, es como deben ser los sistemas para este tipo de tareas.

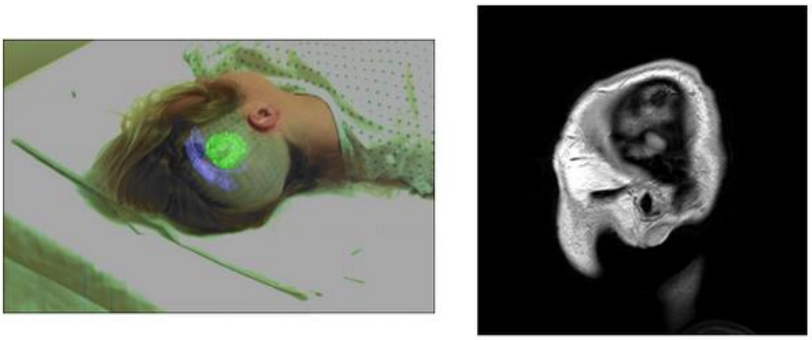


Figura 7: Aplicación médica

Robótica: Quizás la aplicación con la que más encaja este campo de la tecnología, que mejor aplicación para la visión computacional que hacer que los robots vean. Los robots autónomos deben poseer habilidades de procesamiento de datos visuales en tiempo real para poder reaccionar adecuadamente y adaptarse rápidamente a los cambios en el entorno. La detección de objetos fiable y reconocimiento es vital para conseguir este objetivo.

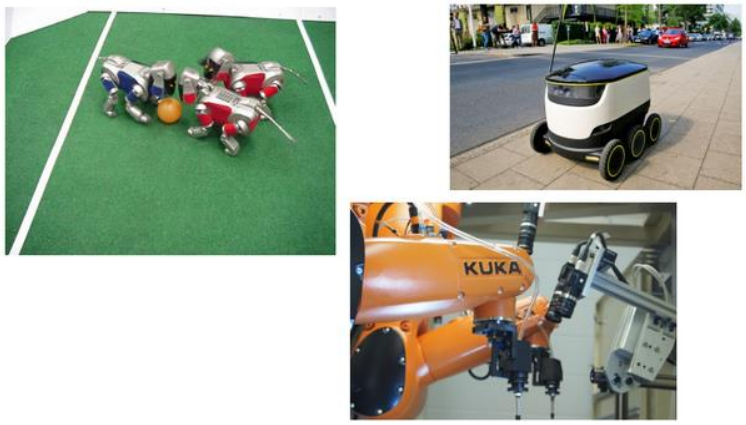


Figura 8: Robótica

Proyecto similar:

El conteo de objetos es otra aplicación muy habitual, hay contadores de múltiples objetos, la mayoría de objetos más cotidianos y de los que hay multitud de bases de datos con imágenes para entrenar modelos. Pero de vez en cuando hay proyectos con características poco comunes.

En el año 2016 por ejemplo se publicó u proyecto titulado *"Automátic detection, traking and counting of birds in marine video content"* [7] publicado por los autores, Roeland T'Jampens, Francisco Hernandez, Florian Vandecasteele and Steven Verstockt.

En esta publicación describen un estudio realizado para el instituto de la marina de Holanda, sobre la construcción de un detector de aves en vuelo en entornos marítimos.

En el artículo describen la dificultas que esta tarea conlleva por la complejidad de los videos tratados, y del acercamiento que se va a tomar hacia la resolución del mismo, realizado mediante una combinación de técnicas de background substraction y detección de objetos basado en redes de Deep learning, para posteriormente realizar un análisis de los patrones de vuelo. En la sección de evaluación del proceso comentan todos los métodos y combinaciones diferentes que probaron antes de llegar a su solución, algoritmos detección de rasgos como SURF, varios de background, y CNN basadas en el conjunto de datos de COCO para la clasificación.

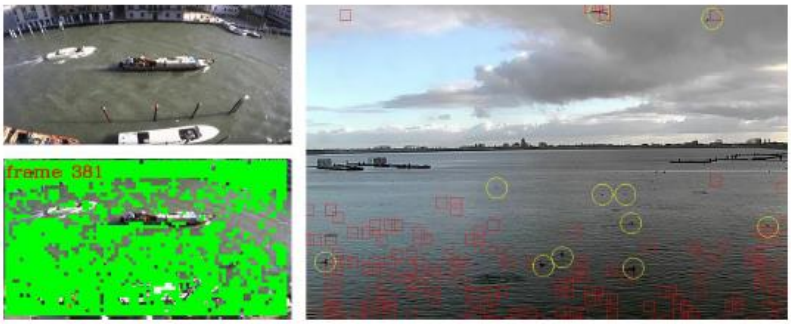


Figura 9: proyecto pájaros 1

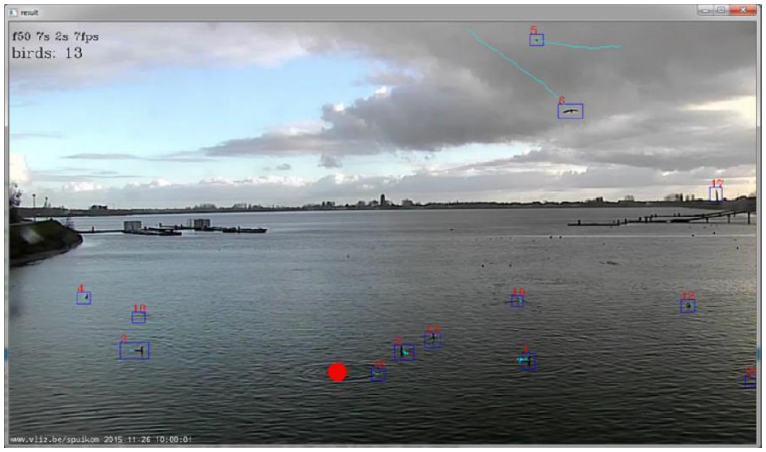


Figura 10: proyecto pájaros 2

El proyecto es muy similar al que se intenta realizar en este documento, pero en él se prueban tanto técnicas tradicionales como las más modernas de Deep learning. Y por tanto la extensión de los algoritmos probados en este es mucho más amplia.

3. ANÁLISIS DEL PROBLEMA

El problema que se busca intentar resolver en este documento es investigar distintos algoritmos de la etapa tradicional de la visión artificial [3] e implementar uno o varios sistemas que haciendo uso de estas técnicas sean capaces de contar el número total de murciélagos volando en las grabaciones nocturnas proporcionadas, con el objetivo de facilitar esta tarea de conteo a los responsables de control de plagas y determinar si es viable emplear algoritmos más modernos en proyectos futuros.

Para lograr este fin se debe analizar el problema que se enfrenta y las tareas que se requieren para resolverla.

El mejor punto para empezar es analizar el dominio para el que se tendrá que diseñar estos sistemas. Se cuenta con un total de 4 videos, en los que se muestran tres diferentes localizaciones de colonias de estos animales. Dos videos muestran la misma localización pero difieren en el flujo de murciélagos en una y en otra.

Las localizaciones son, en orden, el video 1, 2, 3 y 4. Figuras 11, 12 y 13 respectivamente:

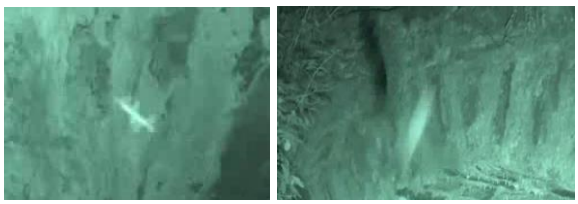
Figura 11: video 1

Figura 12: video 2

Figura 13: videos 3 y 4

Aunque los tres entornos comparten el tipo de grabación y el formato, la cámara apuntado la entrada y salida de la colonia. Difieren en varios aspectos que hay que tener en cuenta, los más llamativos son la forma y tamaño de la entrada y la cantidad de objetos alrededor de cada una que potencialmente pueden dificultar la tarea, objetos que habrá que tener muy en cuenta cuando se diseñen las soluciones. Pero no son las únicas, los factores como la diferencia de iluminación entre ellos o la distancia a la misma también son importantes y se deben tener presentes en la fase de diseño.

También es necesario analizar los objetos de interés en los mismos. Estos son distintas capturas recortadas de los murciélagos con los que se tendrá que tratar en este documento:



En estas imágenes se anticipa que la tarea va a ser complicada, en la mayoría de los casos la forma del murciélago esta apenas insinuada, en otras son meros puntos más luminosos o borrones. El hecho de que las grabaciones sean nocturnas lo complica más todavía ya que no se puede contar con que los colores ayuden a distinguirlos de ninguna manera. Y las grabaciones no tienen buena resolución lo que dificulta la distinción por rasgos concretos.

Visto y examinado el dominio con el que se va a tratar en este proyecto, se necesita responder la pregunta clave. ¿Cómo se puede hacer para contar los murciélagos?

La respuesta a la pregunta es mucho más intuitiva de lo que parece. Como lo haría una persona. Si una persona quisiera contar los murciélagos en los videos, lo primero que haría sería buscarlos en la escena, ver dónde están y cuantos hay. Para asegurarse de que no los cuenta dos veces, la persona los tendría que seguir individualmente a lo largo del video hasta que se perdiesen de vista, una vez fuera del video ya no se puede saber nada de los murciélagos.

Esto es justamente lo que tiene que hacer el sistema que se construya para lidiar con este problema, debe ser capaz de ver a los murciélagos en el video y seguir sus movimientos hasta perderlos de vista cuando ya no se puede saber más de ellos. Estas son las tres partes que se necesitan para satisfacer el problema, encontrar a los murciélagos, seguirlos por el video y contarlos sin repetirlos.

Tras el análisis del problema ya se tiene el conocimiento de con que se está trabajando y que es lo que se necesita hacer. Ahora solo falta el cómo.

4. DISEÑO DE LA SOLUCIÓN

Como ya se ha comentado de pasada en apartados anteriores para este proyecto se van a diseñar cuatro posibles soluciones para el problema que se plantea.

La elección de este número de sistemas se basa en las posibles aproximaciones que se pueden realizar hacia el mismo y en las limitaciones de las herramientas empleadas. Cuando se detectan objetos en videos puede enfocarse de dos maneras. Se puede intentar localizar y reconocer distintas partes del objeto o el propio objeto entero y ser capaz de distinguirlo del resto de elemento a su alrededor. O se puede intentar buscar otra característica que lo haga único además de su forma, como el color o el movimiento.

La idea de diseñar e implementar varios es poder probar las distintas aproximaciones para ver cómo funciona cada una de ellas. No es el objetivo hacer una comparativa entre las mismas, si no ver como de viable es cada una de las soluciones.

Para resolver el problema que se ha planteado, los cuatro sistemas deben estar compuestos por tres partes diferenciadas, a saber: el detector, el tracker o sistema de seguimiento y la función de conteo.

Antes de pasar directamente a las explicaciones, se quiere comentar que no se emplearán ninguna captura de pantalla del código, ya que en general no se recomienda hacerlo y cualquier captura sería poco explicatoria dada la cantidad de funciones específicas y bucles que se emplean.

4.2 Función de Detección

Estas funciones son las encargadas de encontrar al objeto en la imagen y señalar su posición. Para ajustarse a las características del problema descrito, deben ser sistemas lo suficientemente ligeros como para ejecutarse en cada uno de los fotogramas del vídeo.

Son estos los métodos que marcan la distinción entre los acercamientos

4.2.1 Coincidencia de formas (Feature Matching): ORB

Algoritmo

Para entender en que consiste este sistema, primero se debe saber a qué se hace referencia cuando se habla de formas o rasgos, ya que son los elementos más fundamentales e importantes de esta forma de detección.

Rasgos, formas o Features.

No hay una definición universal exacta para decir que es lo que constituye una forma [8], la definición exacta a menudo depende del problema o del tipo de aplicación, pero todas tienen algo en común, y es que son una parte "interesante" de una imagen. Hay varias maneras de referirse a estas formas, otras denominaciones como puntos de interés o rasgos se usan indistintamente.

Si se toma una imagen cualquiera, y se divide en múltiples piezas, como si se tratase de un puzle, cada una contendría formas muy simples, e imaginando que esas piezas se tuviesen que encajar de nuevo, algunas de esas formas serían muy indicativas, y otras no darían nada de información. Mirándolo desde un punto de vista humano para montar un puzle se buscarían patrones como superficies planas, bordes o esquinas, siendo las últimas las más representativas. Para una superficie plana difícilmente se podría localizar su posición, un borde o recta es mucho más descriptivo es fácil aproximar su localización en la imagen aunque la posición exacta sigue planteando un reto, ya que a lo largo, una recta es igual en todas direcciones, es mucho mejor que una zona plana pero deja que desear. Las esquinas sin embargo no tienen estos problemas, son fáciles de localizar, pues no importa cómo se haya cortado, siempre será distinta y única, por eso se considera una buena forma.

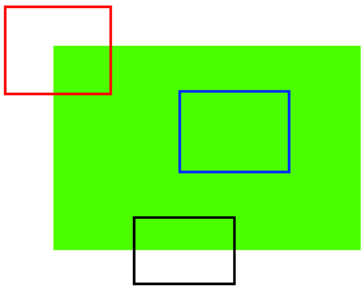


Figura 14: Tipos de rasgos

La imagen de la figura 14, ilustra muy bien lo que se ha explicado, no importa cómo se mueva el rectángulo azul siempre que permanezcas en el interior del cuadrado, todo es igual, el negro sin embargo si puede aportar más información de su posición, pero solo si se mueve en vertical, el movimiento horizontal siempre hará que parezca la misma imagen. El rojo marca la esquina, adonde sea que se mueva siempre tendrá una imagen distinta, por eso se considera a las esquinas como buenos rasgos en una imagen.

Se usan muchos tipos de formas para la detección de objetos [9]. La mayoría están basadas en regiones o en límites de una imagen. Se asume que esta región o límite pertenece a una entidad que es o bien el objeto o parte de un objeto. Algunas de las que se usan se clasifican en:

Formas globales: Suelen ser características de regiones de la imagen tales como el área, tamaño, perímetro, descriptores de Fourier y momentos. Estas formas se pueden obtener considerando todos los puntos de una región o solo aquellos puntos dentro de una zona específica de la región.

Formas Locales: A menudo se encuentran en los límites de un objeto o representados en un área pequeña y distinguible de una región. La curvatura y propiedades similares se usan a menudo. Los puntos de alta curvatura son las ya mencionadas esquinas que juegan el papel más importante de todas en la detección de objetos. Otras también son las mencionadas rectas o segmentos límites.

Formas Relacionales: Las formas relacionales se basan en posiciones relativas de diferentes entidades, ya sean regiones, contornos o formas locales. Estas formas a menudo incluyen la distancia entre formas y medidas de orientación relativas. Son muy útiles para definir la composición de un objeto usando las formas locales de la imagen.

ORB (Oriented FAST and Rotated BRIEF)

Para poder emplear la coincidencia de formas se necesita un algoritmo que haga dos cosas. Una realizada por los denominados **detectores de formas**, estos únicamente se ocupan de encontrar los rasgos en las imágenes, pero lo que se pretende en definitiva es ser capaces de encontrar unos rasgos concretos en otras imágenes, que el ordenador sea capaz de "describir" la región alrededor de la forma para poder encontrarla en otras imágenes y para ello se necesitan algoritmos **de descripción de formas**. Los algoritmos que se emplean en la coincidencia de formas poseen las dos fases la de detección de keypoint, puntos clave, y la fase de descripción.

Dentro de estos algoritmos hay 3 que destacan por su eficiencia y precisión, estos son SIFT, SURF y ORB.

SIFT fue un algoritmo que nació en 2004 de la mano de D.Lowe [10], para dar respuesta al problema de la escalabilidad. No importa cuánto se mueva una forma o rasgo los detectores de esquinas vistos son invariantes ante la rotación es decir que no importan que se rote una esquina que seguirá detectándola, tiene sentido, al fin y al cabo este como esté una esquina sigue siendo eso, una esquina. Sin embargo no ocurre así con la escalabilidad, si se aumenta mucho el tamaño de la imagen pero no el de la ventana una esquina puede dejar de serlo como se ilustra en la figura 15:

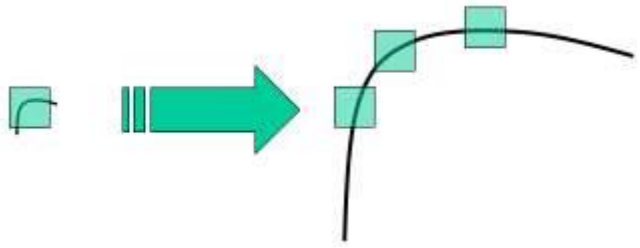


Figura 15: escalabilidad de esquinas

SIFT es un algoritmo creado para superar este problema.

Dos años después apareció SURF otro de los más usados. Nació dada la necesidad de más velocidad a la hora de detectar esquinas, y como su nombre indica (Sepeeded Up Robust Features) no es más que una versión mejorada de SIFT.

Estos dos algoritmos son de los más eficaces dentro del Feature Matching. Sin embargo no se van a manejar en este proyecto, la razón es muy simple, son algoritmos patentados, es decir que para usarlos habría que pagar por ellos, y este proyecto no está financiado de ninguna manera, lo que lleva al primer algoritmo mencionado el ORB [13].

Este es un algoritmo bastante más moderno, apareció en 2011 como una alternativa al SIFT y el SURF en coste computacional, rendimiento y principalmente por las patentes.

ORB es un algoritmo construido sobre el detector FAST y el descriptor BRIEF como indica su nombre (Oriented FAST y Rotated BRIEF), estas técnicas son de reconocido bajo coste y gran rendimiento. Pero para generar un algoritmo lo suficientemente potente como para superar a sus predecesores los creadores de ORB realizaron ciertas adiciones y cambios a los métodos FAST y BRIEF, para entender los cambios se debe entender el funcionamiento de ambos.

FAST. Este algoritmo, aunque no el más preciso si es computacionalmente muy eficiente, por tanto su funcionamiento es muy simple como se muestra a continuación [11]:

Dada la imagen, se selecciona un pixel p para ser identificado como punto de interés o no con una intensidad I_p se selecciona un valor umbral apropiado t y se considera un círculo de 16 píxeles alrededor del pixel p:

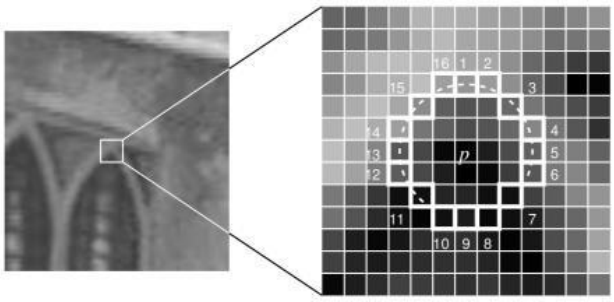


Figura 16: fast 1

Ahora, se considerará al pixel p como una esquina si existe un conjunto de n pixeles contiguos dentro del circulo que sean todos más brillantes que $I_p + t$ o más oscuros que $I_p - t$. En la imagen superior se ha elegido que *n* sea 9.

Se propuso emplear un test de alta velocidad para descartar grandes números de pixeles que no eran una esquina. En el test solo se examinaban los cuatro píxeles en las posiciones 1, 9, 5, 13, si 1 y 9 eran más luminosos o más oscuros que *p*, entonces se examina el 5 y 13. Si p resulta ser una esquina entonces es que al menos tres de estos cuatro números deben ser más luminosos o más oscuros. Si no se da ninguno de estos casos entonces p no puede ser una esquina. Esta forma de detección tiene buen rendimiento, pero también varias debilidades, a saber: no rechaza tantos candidatos cuando se emplea $n < 12$, la elección de los pixeles no es óptima porque su eficiencia depende del orden de la cuestión y distribución de las aparentes esquinas, los resultados del test de gran velocidad no se guardan sino que se descartan y da problemas cuando hay muchos rasgos adyacentes.

Los tres primeros problemas se solventan con una aproximación al aprendizaje automático. Se selecciona un set de imágenes para el entrenamiento y se corre el algoritmo FAST en todas las imágenes. Para cada punto *p* se guardan los 16 pixeles a si alrededor para obtener así el vector de formas *P*. Cada pixel *x* de entre estos 16 puede tener tres estados:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t & \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t & \text{(similar)} \\ b, & I_p + t \leq I_{p\to x} & \text{(brighter)} \end{cases}$$

Dependiendo de estos tres estados (mas oscuro, similar, mas luminoso), el vector P se subdivide en otros tres, P_d, P_s, P_b. Se define una nueva variable booleana, K_p que se define como verdadera si p es esquina y falsa en caso contrario. Se usa el algoritmo ID3 (un árbol de decisión para clasificación) y para buscar cada subset empleando la variable K_p . Se selecciona el *x* que más información aporta sobre si el pixel es o no una esquina medido mediante la entropía de K_p . Esto se aplica recursivamente hasta que la entropía es cero. El árbol de decisión que se crea se emplea para detección rápida en otras imágenes.

Ahora el problema de los múltiples rasgos adyacentes se debe solucionar aplicando la supresión no máxima (Non-maximal suppression). Se procesa una función de puntuación *V* para todos los rasgos detectados. *V* es la suma absoluta de las diferencia entre el valor de *p* y los 16 pixeles que le rodea. Se toman dos rasgos adyacentes y se procesan sus funciones *V* y se descarta aquel que tenga el valor más bajo.

Ahora, cuando se emplea en ORB se cambian varias cosas.

1. Se usa un radio de 9 píxeles en lugar de 16 ya que tiene mejor rendimiento, a menos pixeles menos operaciones.
2. Puesto que FAST no produce ninguna medida de importancia y se sabe que es habitual detectar los bordes además de las esquinas, se emplea la medida para esquinas de Harris para ordenar estos puntos clave que detecta FAST. Para un numero *N* de rasgos deseados, se establece un umbral lo suficientemente bajo como para detectar un número mayor de rasgos que *N*, se ordenan con Harris y se cogen los *N* primeros.
3. Como FAST también sufre el problema del multi escalado, se emplea una pirámide de escalas en la imagen y se realiza generan rasgos sobre cada nivel de la pirámide.

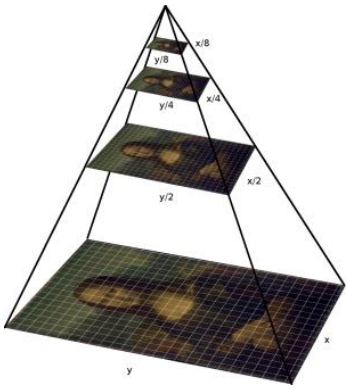


Figura 17: fast pirámide

Ahora otro problema, FAST no es un algoritmo que aplique orientación, hacia donde está orientada esa esquina. Para poder obtenerla ORB aplica un algoritmo conocido como intensidad del centroide. Este algoritmo asume que la intensidad de la esquina está separada de su centro y que el vector resultante se puede emplear para obtener la orientación.

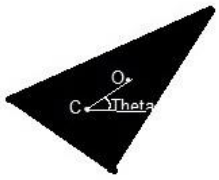


Figura 18: fast orientación

Se define el momento como:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

Y con este momento se puede obtener el centroide:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

Y se puede construir el vector con el centro de la esquina *O* al centroide –*OC*. Se obtiene así la orientación dada por:

$$\theta = \text{atan2}(m_{01}, m_{10})$$

Una vez calculada la orientación se puede rotar con una rotación canónica y pasar al descriptor, así se obtiene además una invariancia ante la rotación.

BRIEF (Binary Robust Independent Elementary Feature). Es un descriptor [12], que en este caso toma todos los puntos clave detectados por FAST y los convierte en un vector binario de rasgos de tal forma que juntos representen un objeto. Estos vectores, también conocidos como descriptores de rasgos binarios son vectores que solo contienen 0 y 1. En

resumen cada punto clave está representado por uno de estos vectores de entre 128-512 bits.

BRIEF empieza por realizar un suavizado a la imagen mediante un kernel Gaussiano para evitar que el descriptor sea sensible a los ruidos. Después se selecciona un par de pixeles en un vecindario definido alrededor del punto de interés, este vecindario se denomina como parche, un cuadrado con un alto y un ancho. El primer pixel entre el par aleatorio se extrae de una distribución gaussiana centrada alrededor del punto de interés con una desviación estándar. El segundo pixel del par se extrae de una distribución gaussiana centrada alrededor del primer pixel con el doble de desviación estándar. Si el primer pixel del par es más luminoso que el segundo se le asigna el valor 1, en caso contrario el 0. Ilustrado en la figura 19.

Figura 19: BRIEF suavizado

De nuevo se selecciona un par aleatorio y se les asignan los valores que les correspondan. Para un vector de 128 bits de extensión, se repite este proceso 128 veces. BRIEF crea un vector como este para cada uno de los puntos de interés en una imagen.

Sin embargo hay un problema adicional, y es que BRIEF no es invariante ante la rotación. Para añadir esta funcionalidad, ORB aplica un método que no repercute demasiado en el aspecto veloz de BRIEF.

Considérese el parche de una imagen suavizada como *p*. El test binario τ se define por:

$$\begin{aligned} &Where\,\tau(p;x,y)\,is\,defined\,as: \\ &\tau(p;x,y) = \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) \geq p(y) \end{cases} \\ &p(x)\,is\,the\,\text{int}ensity\,value\,at\,pixel\,x. \end{aligned}$$

Donde *p(x)* es la intensidad de *p* en el punto *x*. Como vector el rasgo se define como:

$$f(n) = \sum_{1<i<n} 2^{i-1}\tau(p;x_i,y_i)$$

El rendimiento de coincidencia de BRIEF cae enormemente cuando se aplica una rotación de más de unos pocos grados. En ORB se dirige BRIEF de acuerdo a la orientación de los puntos clave. Para todo rasgo de *n* tests binarios en la posición (x_i , y_i) se necesita una matriz de tamaño 2 X n:

$$S = \begin{pmatrix} x1,\ldots x_n \\ y1,\ldots y_n \end{pmatrix}$$

Se usa un parche con una orientación θ y la correspondiente matriz de rotación R_θ y se aplica sobre la matriz *S*:

$$S_\theta = R_\theta S$$

De tal forma que el operador BRIEF dirigido se convierte en:

$$g_n(p,\theta) = f_n(p) \mid (x_i, y_i) \in S_\theta$$

Con esta ecuación, mientras la orientación θ sea consistente a lo largo de las vistas, se usará S_θ para procesar el descriptor.

ORB emplea el denominado rBRIEF de la siguiente manera:

1. Corre cada test sobre todos los parches de entrenamiento
2. Ordena los test por su distancia de la media 0.5 formando el vector *T*
3. Realiza una búsqueda avariciosa:
    a. Pone el primer test en el vector de resultados *R* y lo quita de *T*
    b. Coge el siguiente test de *T* y lo compara con todos los tests en *R*, si la correlación absoluta es mayor que un umbral, se descarta, en caso contrario se añade a *R*.
    c. Repetir los pasos anteriores hasta que haya 256 tests en *R*. Si hay menos de 256 se quita el umbral y se prueba de nuevo.

El ensamblado de estos dos da lugar al algoritmo de ORB.

Implementación

Como ya se ha explicado, ORB es un algoritmo que busca rasgos descriptivos en una imagen que suelen señalar la presencia de un objeto, tales como esquinas o bordes.

Lo primero antes de empezar la detección, es darle al algoritmo lo que precisa para poder encontrar a los murciélagos en el video, se necesita que ORB primero detecte los rasgos en una imagen modelo de un murciélago. Se ha escogido una diferente para cada video, buscando que sean lo más nítidas posibles para poder encontrar cuantos más rasgos mejor.

Una vez escogida la imagen se procede a crear el propio algoritmo de ORB. Este tiene multitud de parámetros que se deben ajustar para que este lo más orientado posible al tipo de objeto a buscar.

Lo primero que se debe decidir es el número de rasgos que el algoritmo marcará en un mismo fotograma, estos puntos estarán jerarquizados, es decir, si se ponen diez rasgos, se marcarán los diez considerados por ORB como más relevantes o pronunciados. Se tienen

que tener en cuenta factores como el ruido, el número de murciélagos en pantalla o la nitidez y cercanía de los mismos. Se busca el equilibrio entre encontrar murciélagos y no captar otras cosas. Los murciélagos en los videos son en la mayoría de los casos muy pequeños y no es posible extraer demasiados rasgos de ellos, pero hay que asegurarse de que los rasgos marcados no se gasten en los objetos que no son murciélagos, 150 es el número que se ha decidido.

Con anterioridad ya se explicó que ORB emplea un sistema de pirámides para escalar las imágenes. Este sistema de pirámides emplea un factor de escala, un número mayor que 1, que indica el radio de reducción de la pirámide. Un valor dos implica que cada nivel será 4x pixeles menor que el nivel anterior. Un factor de escala muy grande degradaría la coincidencia, pero uno muy pequeño implica que para ciertos rangos de escala se necesitarían muchos niveles lo que empeoraría la velocidad. Se busca que el detector sea rápido, pero es más importante que sea preciso, 1.2 de escala es apropiado para problemas donde se buscan rasgos muy pequeños. Lo siguiente claro está es determinar el tamaño de la pirámide, cuantos niveles tendrá esta, sabiendo que el nivel más pequeño tendrá un tamaño lineal igual al de la imagen de entrada. Puesto que el factor de escala es bastante bajo, no es necesario un gran tamaño, 10 se ha probado correcto, apoyándose en otros proyectos similares. También de forma opcional se puede escoger en qué nivel se quiere introducir la imagen de entrada, por defecto es 0, la primera y a sí se ha dejado, no hay motivos para cambiarlo.

Se establece un valor umbral que marca el tamaño de los rasgos a partir del cual estos no serán detectados. Este valor tiene que estar en concordancia con el tamaño de la ventana que como ya se explicó emplea BRIEF para determinar el vecindario alrededor de un punto de interés, tamaño que también hay que determinar. Ya se ha dicho que las figuras que se buscan son bastante pequeñas, con lo que no hace falta detectar formas muy grandes, un valor que asegure incluir las formas que se busquen y limite un poco las que no es lo que hace falta. Se estimó un valor de 17 para ambos ya que tienen que ir acorde.

Ahora se debe establecer el número de puntos que genera BRIEF por cada elemento. Se recuerda que el algoritmo de BRIEF por defecto selecciona dos pixeles al azar dentro del vecindario de un posible punto de interés. Este valor permite elegir si se quiere mantener en 2 o se quiere que sean 3 o 4. En principio cuantos más puntos más preciso, pero eso también implica que hay más posibilidades de que rechace un punto como punto de interés. En esta ocasión se elige la inclusión a la precisión y se mantiene el valor 2.

Por último hay que determinar mediante que algoritmo se valorará que un rasgo sea más importante que el otro. Hay dos métodos, el de HARRIS y el FAST_SCORE. El segundo es más rápido y el primero más preciso. Aquí se escoge la precisión ya que la velocidad de esto no es muy notable, se elige HARRIS.

Ya está calibrado y creado el ORB que se va a emplear, pero ORB es únicamente el que detecta los rasgos y los describe, se necesita hacerlas coincidir con rasgos similares o iguales, aquí entran en juego los Matchers. Estos algoritmos son los que comparan los rasgos detectados de la imagen modelo con los rasgos detectados de la imagen objetivo.

Hay dos, uno muy simple que es literalmente un matcher de fuerza bruta, que va comparando uno a uno los rasgos de ambas imágenes empleando cálculos de distancias, y los más cercanos se devuelven. O el matcher basado en FLANN, este último se sirve de una serie de algoritmos que optimizan una búsqueda del tipo vecino más cercano. Es más rápido que el de fuerza bruta, especialmente para grandes conjuntos de datos y es igual de preciso. Este es el que se empleará en la implementación. Requiere de dos diccionarios para emplearse. Para filtrar los matches este algoritmo emplea un test de la distancia de radio para eliminar las coincidencias falsas. Los diccionarios que se le pasan deben contener datos de carácter un poco técnico, especialmente el primero cuyos datos sirven para configurar los arboles de decisión que FLANN construirá sobre el problema, los valores por defecto se recomiendan no modificarlos, tras algunas pruebas se decidió que era mejor dejarlos así. El segundo diccionario contiene el valor que determinará el número de veces que el árbol será recorrido, cuantas más veces más precisión, pero más tiempo de cómputo. 500 fue el valor más alto que se probó que no afectaba visiblemente al rendimiento.

Con las dos partes del componente creadas el detector ya está casi listo, solo falta ejecutarlo sobre cada fotograma y que genere unas coordenadas para pasárselo al tracker.

Se genera un bucle que pasa los fotogramas del video uno cada ejecución, este fotograma se pasa a una escala de colores grises para detectar las intensidades más fácilmente, esta imagen gris se procesa con ORB como se hizo con la imagen modelo, y sobre las dos imágenes computadas se ejecuta el Flann, este genera una variable de salida que se denomina como matches, este valor es un array que contiene todos los puntos considerados buenos matches. Sobre estos maches se realizan transformaciones de los valores hasta obtener una serie de coordenadas sobre los cuales se dibuja una máscara que si los matches no están demasiado separados entre sí tendrá forma rectangular, si están demasiado lejos, la máscara tomará una forma un poco más abstracta.

Las coordenadas de esta máscara es lo que se le pasa al tracker, coordenadas que señalan la posición en la que supuestamente se encuentra el objeto.

4.2.2 Aprendizaje automático: Haar-Cascade

Algoritmo

El algoritmo de Haar cascade es un algoritmo de detección de objetos basado en el aprendizaje automático. Fue propuesto en 2001 como ya se ha mencionado, por Paul Viola y Michael Jones [14].

Esta aproximación se basa en una función en cascada que se entrena a partir de multitud de imágenes positivas y negativas.

El algoritmo tiene 4 estados:

1. Selección de rasgos de Haar
2. Creación de imágenes integrales
3. Entrenamiento de Adaboost

4. Clasificadores en cascada.

Este algoritmo funciona con regiones rectangulares adyacentes en una localización específica en la ventana de detección, suma las intensidades de pixel en cada región y calcula la diferencia entre esas sumas. Figura 20.

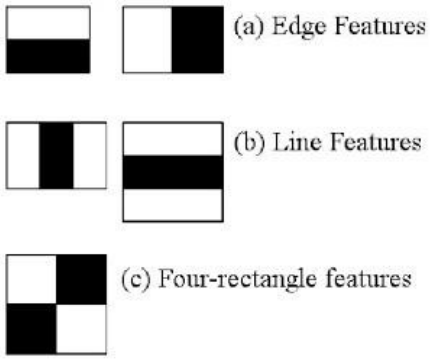


Figura 20: Haar intensidades

Pero de entre todas esas figuras que se calculan, la mayoría son irrelevantes. A sí que para seleccionar los rasgos correctos, se emplea el Adaboost que selecciona las mejores forma y entrena a los clasificadores que las usan: Este algoritmo construye un fuerte clasificador mediante una combinación lineal de clasificadores más débiles.

El proceso comienza en la fase de detección, la ventana objetivo se mueve por la imagen de entrada y por cada subsección de la imagen se calculan los rasgos de Haar. Las diferencias de intensidades se comparan con un umbral conocido que separa los objetos de los no objetos. Puesto que Haar es solo un clasificador débil, se necesita una gran cantidad de rasgos de Haar para describir un objeto con la suficiente precisión y organizarlos en clasificadores en cascada para formar un clasificador robusto.

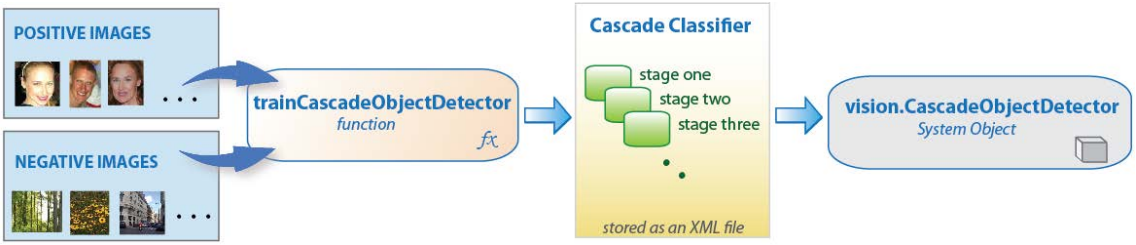


Figura 21: flujo de Haar

El clasificador en cascada consta de una serie de estados, como se muestra en la figura 21, donde cada uno es un conjunto de clasificadores débiles. Estos clasificadores se denominan pasos de decisión. Cada uno se entrena con la técnica de boosting, que permite entrenar clasificadores muy precisos empleando el peso medio de las decisiones tomadas por los clasificadores débiles.

Cada estado de clasificación etiqueta la región observada como positiva o negativa. Positiva cuando se encuentra el objeto, negativa cuando no. Si la etiqueta es negativa, la clasificación de la región esta completada, y se pasa a la siguiente. Si es positiva el

clasificador pasa la zona al siguiente estado. Cuando en el último de los estados se declara que la región es positiva, entonces el detector lo marca como objeto.

Estos estados están creados para rechazar muestras negativas lo más rápido posible. La suposición es que la gran mayoría de las ventanas no contienen objetos de interés. Para que funcionen bien, cada estado está configurado para detectar pocos falsos negativos, pues una vez se etiqueta un objeto como negativo, ya no se puede deshacer. Cosa que no ocurre cuando se detecta un no objeto como positivo, los estados siguientes pueden corregirlo. Cuantos más estados más se reduce la probabilidad de detectar un objeto como no objeto, pero aumentan las inversas.

Implementación

La ventaja de este método es que mediante librerías es cómodo de emplear en OpenCV. El mayor trabajo que lleva, es también el más vital, entrenar un modelo. En internet hay varios modelos preentrenados para emplear con el Haar, todos para objetos relativamente comunes, rostros, personas, coches, etc… seguramente haya algún modelo entrenado para captar murciélagos, pero no como los que se emplean en este trabajo, se trata de modelos entrenados para imágenes nítidas, como las que se encuentran buscando en internet. No, las imágenes que se tratan en este modelo son imágenes únicas, si se desea crear un detector con Haar-cascade, se tiene que entrenar un modelo.

Para entrenar el modelo se necesitan dos cosas, imágenes positivas e imágenes negativas, siendo las positivas imágenes que contienen el objeto deseado señalado y las negativas imágenes de fondos en blanco y negro, imágenes que no contienen el objeto deseado y que servirán para enseñarle al modelo que no mirar cuando esté buscando, estas últimas si se han obtenido de internet donde hay multitud de bancos gratuitos. Tanto las imágenes negativas como las positivas deben ser registradas en un archivo de descripción que emplearán los métodos de generación del modelo.

Con las imágenes positivas se hacen varias cosas, la primera es que hay que marcar manualmente los objetos que se desean en imágenes que en este caso se han obtenido directamente de los videos en los que se van a probar, lo ideal es que estas muestras provengan de entornos lo más variados posibles y no directamente de los videos objetivo, pero no hay otra forma de extraer las imágenes que se necesitan. Con las imágenes transformadas en archivos bmp, mapas de bits, necesario por el modo en el que este formato representa la información de una imagen y las coordenadas de los objetos de interés señalados en un archivo de descripción (fácil de generar con herramientas gratuitas de internet).

OpenCV también incluye varias funciones para entrenar los modelos de Haar. Habiendo realizado la separación de muestras, y habiendo generado los archivos de referencia para cada uno de ellos solo se deben pasar estos datos a los métodos de entrenamiento, y de esta forma obtener un modelo de tipo .xml con el que se ejecutará el método de Haar-cascade.

El método de entrenamiento de cascada pide entre otras cosas el número de estados que se desean crear, como ya se explicó en la sección anterior, cuantos más estados más

seguro es que el objeto que los pasa todos es realmente el que se está buscando, pero al mismo tiempo es más probable que objetos que lo son sean rechazados. En vista de que en el problema que se trabaja la mayoría de las veces los murciélagos no tienen una forma definida, sino que son poco más que manchas o borrones, se busca que haya suficientes estados para el tamaño de entrada, pero que aunque se dejen pasar objetos que no lo son, que no se pierdan los que sí. Para el tamaño de imágenes positivas 447, se estableció un número de 14 estados.

Ya se tiene el modelo, ahora solo se tiene que implementar en el algoritmo de Haar. Se llama al método de clasificador en cascada con este modelo como parámetro de entrada. Se recorre el video con un bucle que va fotograma a fotograma hasta que termina. A cada fotograma se le aplica una transformación de tonalidad, se les transforma a imágenes en tonos grises, esto ya se hizo para la coincidencia de formas y se volverá a hacer en las técnicas de background, el motivo es que una imagen en tonos grises es mucho más fácil de manejar y mucho más efectiva cuando se trabaja con las intensidades de los píxeles, ya que la única diferencia entre un pixel u otro es lo oscuro o claro que estos son. Como ya se comentó cuando se explicó el algoritmo, este toma decisiones basado en las intensidades, de ahí la conversión a una imagen en tonos grises.

Ya sobre esta imagen se aplica el clasificador en cascada con un detector de multi escala. Esto genera un array que contiene las coordenadas de los objetos que el clasificador ha encontrado en la imagen, con estas coordenadas se pinta la caja delimitadora y se pasa al tracker.

4.2.3 Background Substraction: MOG2 y GMG

Background substraction es uno de los métodos más efectivos para identificar objetos, no por como son sino por su característica de movimiento en una secuencia de video y es de hecho un paso fundamental en muchos sistemas de visión artificial y de sistemas de seguridad o de tráfico. Básicamente estos algoritmos de substracción necesitan de un fondo estable, lo que suele resultar algo muy complicado en aplicaciones de la vida real. En este método las secuencias de video se dividen en frames (fotogramas) y cada frame se emplea como referencia para crear un modelo del fondo. Los píxeles del frame actual, que difieran del modelo del fondo se consideran objetos en movimiento. Para el uso en localización de objetos y seguimiento de los mismos, se necesita realizar más procesamiento sobre estos objetos denominados de foreground (primer plano). Dado que la extracción del fondo suele ser el primer paso en cualquier aplicación de visión artificial, es crucial que los píxeles extraídos del primer plano correspondan correctamente con el objeto en movimiento de interés.

La técnica de background substraction es un algoritmo muy simple pero muy sensible ante cambios externos en el ambiente, y se ve muy afectado por las interferencias. Este método puede dar la información casi perfecta del objeto que se busca si se posee el fondo integro.

Hay muchos desafíos a la hora de desarrollar un buen algoritmo de extracción de fondo, el algoritmo debe ser robusto ante cambios en la iluminación, debería ser capaz de evitar

detectar partes del fondo que no sean estacionarias, como hojas, lluvia, sombras, etc… y el modelo del fondo interno debería cambiar rápidamente si hay algún cambio en el fondo.

Algoritmo base:

Muchos algoritmos de extracción de fondo han sido propuestos en años recientes, pero identificar objetos móviles en un fondo complejo sigue siendo un reto. La mayoría de los algoritmos siguen el mismo modelo básico consistente en cuatro pasos principales; preprocesamiento (pre-processing), modelado del fondo (Background Modelling), detección del primer plano (Foreground detection) y validación de los datos (Data Validation). La imagen de la figura 22 lo ilustra:

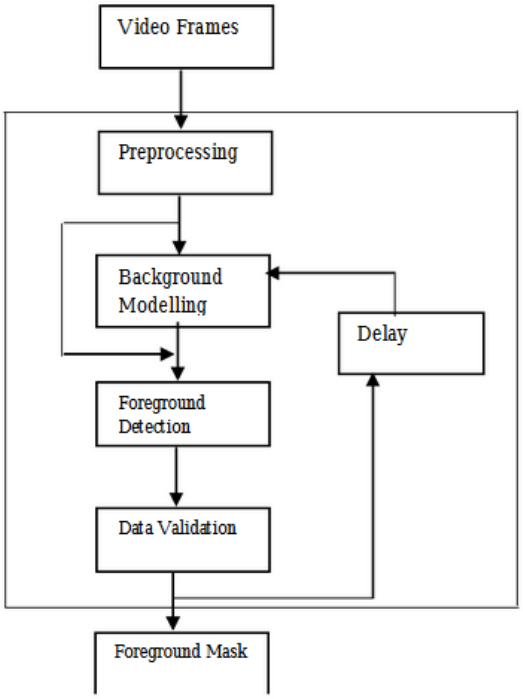


Figura 22: flujo de Background

- **Preprocesamiento:** Este paso cambia la entrada del video base en un formato que puede ser procesado fácilmente por los pasos posteriores del algoritmo. Consiste en una serie de tareas de procesamiento de imágenes tales como eliminar ruido, suavizar la imagen y eliminar el ruido ambiental transitorio. Este paso de eliminación de ruido se refiere a la captura de lluvia, nieve y similares por parte de la cámara. Para disminuir la cantidad de datos a procesar se emplean reducciones del tamaño y del número de fotogramas por segundo (frame rate). En el caso de múltiples cámaras, antes del modelado del fondo se necesita realizar un registro de imágenes entre las diferentes cámaras.
- **Modelado del fondo:** Este es uno de los pasos más importantes en este tipo de algoritmos. El principal objetivo de este modelado es conseguir capturar información sobre la secuencia de fotogramas del vídeo y actualizar la información para cambiar el escenario del fondo si es necesario. Para detectar el plano frontal es fundamental modelar el fondo,

a lo largo de los años se han propuesto varias técnicas para conseguirlo, la mayoría de las cuales siguen el mismo esquema.

El primer fotograma o el previo se usan para construir el modelo del fondo, entonces se compara este modelo con el frame que se esté manejando en el momento para detectar los objetos en el plano frontal, y entonces se actualiza el modelo del fondo. Este modelado se divide en dos métodos, los métodos paramétricos y los no paramétricos.

Los modelos paramétricos usan un modelo estadístico adaptativo para detectar cambios en la escena del video usando un modelo de Gauss multidimensional. El valor del pixel en el fotograma actual se compara con el modelo del fondo con el objetivo de clasificar los pixeles como fondo.

La principal diferencia de los modelos no paramétricos es que estos tratan de modelar el fondo con gran precisión, estos modelos se adaptan más rápidamente a los cambios en el fondo y ser capaces de detectar objetivos con gran sensibilidad para lo cual capturan la información más reciente sobre la secuencia del video y actualizan constantemente la información.

- **Detección del plano frontal:** En este paso los fotogramas del video de entrada se comparan con el modelo del fondo en el paso anterior para identificar aquellos pixeles que difieren y marcarlos como objetos del primer plano. Una de las aproximaciones más comunes para este tipo de detección es comprobar si el pixel en el fotograma de entrada es significativamente diferente del pixel correspondiente en el modelo del fondo.
- **Validación de los datos:** Este paso es para mejorar el candidato a pertenecer a la máscara frontal basándose en la información obtenida fuera del modelo del fondo.

Aunque la mayoría de los algoritmos siguen este esquema no todos emplean las mismas técnicas en cada fase. Hay muchos acercamientos posibles, algunos más simples con la intención de maximizar la velocidad y limitar los requerimientos de memoria, hasta otros más sofisticados que intentan alcanzar la máxima precisión posible bajo cualquier circunstancia. En orden de complejidad de menor a mayor estos son algunos de los posibles acercamientos:

Ejecución de la media Gaussiana, filtro de la media temporal, modelo de mezcla Gaussiana (GMM), estimación de la densidad del kernel (KDE), aproximación secuencial de la densidad del kernel, variación de la concurrencia de imágenes o fondo de Eigen.

En este proyecto se explorara el funcionamiento de aquellos métodos implementados mediante librerías en la herramienta visual de OpenCV, puesto que la implementación manual de cada uno de ellos es extremadamente compleja y excede los objetivos de este proyecto. Además ambos son métodos de probado rendimiento y eficacia. Estos dos algoritmos son background substractor MOG2 y background substractor GMG.

MOG2 (Mixture Of Gaussians)

Lo primero e mencionar es que este algoritmo es una versión mejorada del algoritmo original MOG, su funcionamiento es prácticamente el mismo excepto por una función que se señalará cuando se explique. Como su nombre indica utiliza un modelo de segmentación de fondo /frente (background/foreground) basado en un modelo de mezclas gaussianas [15].

Cuando se busca detectar movimiento en usa secuencia de video, una suposición que se asume muy a menudo es que las imágenes de una escena sin los objetos considerados intrusos, exhiben un comportamiento regular que puede ser bien descrito por un modelo estadístico. Si se posee este modelo estadístico de la escena cualquier objeto intruso que aparezca en la misma puede ser detectado observando las partes de la imagen que no encajan con el modelo. Una aproximación habitual es aplicar una función de probabilidad de densidad para cada pixel individual del modelo de la escena. Un pixel de una nueva imagen es considerado perteneciente al fondo si su valor está bien descrito por la su función de densidad.

GMM (Gaussian Mixture Model), es un modelo probabilístico que representa la distribución normal de subpoblaciones dentro de una población mayor. Este tipo de modelos no requieren conocer a que subpoblación pertenece un dato, permitiendo que el modelo aprenda de las subpoblaciones automáticamente como si de un set de entrenamiento se tratase, esto constituye una forma de aprendizaje no supervisado. El GMM está parametrizado por dos tipos de valores los pesos de los componentes y las medias, y las covarianzas.

Este principio de las GMM se aplica a las imágenes considerando el conjunto total de los pixeles como la población total y los que pertenecen al fondo o al frente como las subpoblaciones.

Las matemáticas del modelo aplicado a las imágenes son bastante complejas, pero entenderlas es necesario para comparar su funcionamiento al de otros modelos. Se intentaran explicar lo más simplificado posible.

El modelo de GMM que se propone en este algoritmo de MOG2 se basa en que puesto que la iluminación de una escena puede cambiar por múltiples motivos, la hora del día, el tiempo atmosférico, la luz de una bombilla que se enciende o el flash de una cámara, un nuevo objeto puede entrar en la escena o uno que estaba presente puede desaparecer. Para poder adaptarse a estos cambios se debe actualizar el set de entrenamiento añadiendo nuevas muestras y descartando las antiguas. Para ello establece un periodo de tiempo razonable *T* y en un instante t se tiene $\mathcal{X}_T = \{x^{(t)}, ..., x^{(t-T)}\}$. Para cada nueva muestra se actualiza el set de entrenamiento X_T y se recalcula la probabilidad $\mathcal{X}_T = \{x^{(t)}, ..., x^{(t-T)}\}$. Sin embargo de entre el historial de las muestras más recientes puede haber valores que pertenezcan a los objetos del primer plano y esta estimación se denota como $p(\vec{x}^{(t)}|\mathcal{X}_T, B\vec{G}+FG)$. Se emplea la GMM con *M* componentes:

$$\hat{p}(\vec{x}|\mathcal{X}_T, BG+FG) = \sum_{m=1}^{M} \hat{\pi}_m \mathcal{N}(\vec{x}; \widehat{\vec{\mu}}_m, \widehat{\sigma}_m^2 I)$$

Donde $\widehat{\vec{\mu}}_1, \ldots, \widehat{\vec{\mu}}_M$ son las estimaciones de las medias y $\widehat{\sigma}_1, \ldots, \widehat{\sigma}_M$ las de las varianzas que describen los componentes de Gauss. Se asume que las matrices de covarianza son diagonales y la identidad de la matriz *I* tiene dimensiones adecuadas. Los pesos de la mezcla $\hat{\pi}_m$, son no negativos y suman hasta uno. Dada una nueva muestra $\vec{x}^{(t)}$ en el instante *t* las ecuaciones de actualización recursivas son en orden por línea (1), (2), (3):

$$\begin{aligned}
\hat{\pi}_m &\leftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m) \\
\widehat{\vec{\mu}}_m &\leftarrow \widehat{\vec{\mu}}_m + o_m^{(t)}(\alpha/\hat{\pi}_m)\vec{\delta}_m \\
\widehat{\sigma}_m^2 &\leftarrow \widehat{\sigma}_m^2 + o_m^{(t)}(\alpha/\hat{\pi}_m)(\vec{\delta}_m^T\vec{\delta}_m - \widehat{\sigma}_m^2),
\end{aligned}$$

Donde $\vec{\delta}_m = \vec{x}^{(t)} - \widehat{\vec{\mu}}_m$ y en lugar del intervalo *T* mencionado al principio, aquí la constante α describe la decadencia exponencial que se emplea para limitar la influencia de los datos más antiguos. Este valor es aproximadamente *1/T*.

Habitualmente el objeto intruso se representa con algunos clusters adicionales unos pesos pequeños: Y por tanto se puede aproximar el modelo del fondo como:

$$p(\vec{x}|\mathcal{X}_T, BG) \sim \sum_{m=1}^{B} \hat{\pi}_m \mathcal{N}(\vec{x}; \widehat{\vec{\mu}}_m, \sigma_m^2 I)$$

Si los componentes están ordenados para tener los pesos descendientes, entonces se puede sacar:

$$B = \arg\min_b \left(\sum_{m=1}^{b} \hat{\pi}_m > (1 - c_f) \right)$$

Donde c_f es una medida de la porción máxima de datos que pueden pertenecer a la parte frontal sin influenciar en los objetos del fondo. Por ejemplo si un nuevo objeto aparece en la escena y se mantiene estático durante un rato, probablemente genere varios clusters adicionales. Puesto que el antiguo fondo esta opacado, los pesos del nuevo cluster se verán incrementados constantemente. Si el objeto se mantiene estático el tiempo suficiente y su peso se vuelve más grande que c_f entonces puede ser considerado como parte del modelo del fondo.

Pero aún queda saber cómo se decide el número de componentes, o de distribuciones gaussianas que se aplican a cada pixel. Este paso es la mayor diferencia entre el modelo de MOG y el MOG2. En el primero se emplean un número de distribuciones de 3 a 5, sin embargo en el modelo más avanzado de MOG2 se selecciona individualmente el número apropiado de distribuciones para cada pixel.

Esto se obtiene partiendo del peso π_m, que es la fracción del dato que pertenece al componente m del GMM. Este puede ser considerado como la probabilidad de que una muestra provenga del componente m y de esta manera π_m-S define una distribución multinomial subyacente. Con esto se puede asumir que teniendo *t* muestras de datos, cada una pertenece a uno de los componentes de la GMM. Siguiendo esta línea de pensamiento, y desarrollándolo, se llega a la ecuación:

$$\hat{\pi}_m \leftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m) - \alpha c_T.$$

Que en este modelo MOG2 sustituye a la ecuación primera de las tres que representan la actualización recursiva.

Después de cada actualización se va a tener que normalizar el valor de π_m-S para que sumen hasta 1. Se empieza con un GMM con un componente centrado en la primera muestra y los nuevos componentes se van añadiendo según toque. Un peso negativo de Dirichlet prior eliminará los componentes que no estén apoyados por los datos y entonces se podrán descartar los componentes *m* cuando su peso se vuelva negativo.

De esta manera es como se obtiene un modelo del fondo para realizar la segmentación con MOG2.

GMG

Este es un sistema creado para funcionar en tiempo real, unos 15 fotogramas por segundo y no requiere de ningún entrenamiento para funcionar. El método [16] se divide en dos partes:

La primera es un algoritmo de segmentación probabilístico del primer plano que identifica los posibles objetos que pertenecen al plano frontal usando inferencia bayesiana con un modelo de estimador de fondo tiempo variante y un modelo de primer plano inferido. El modelo del fondo consiste en una distribución no paramétrica en métrica de colores RGB para cada pixel en la imagen o fotograma. Los estimadores son adaptativos, y a las nuevas observaciones se les dan más peso que a las antiguas para ajustarse a la iluminación variable.

La segunda parte es un sistema de seguimiento múltiple, que filtra selectivamente y refina los objetos propuestos del primer plano. El filtrado selectivo se logra con un modelo de confianza heurístico que incorpora un error de covarianza calculado por el algoritmo de seguimiento múltiple. Este sistema de seguimiento fue implementado por los creadores de este detector con el objetivo final de ilustrar el funcionamiento de GMG siguiendo personas en distintos videos, puesto que esta parte no está implementada en OpenCV y de hecho esta orientación no es la que interesa en este proyecto se prescindirá de este sistema de seguimiento pre configurado para personas.

Las matemáticas que se aplican son bastante complejas y largas, se intentara dar una explicación detallada de los pasos sin desarrollar en exceso las ecuaciones matemáticas.

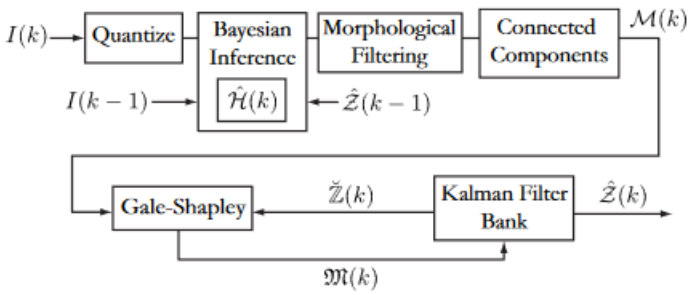


Figura 23: GMG proceso

Segmentación probabilística del plano frontal: Esta primera parte, la que se estudiará en este documento se centra en la primera línea de la figura 23, donde se obtiene una imagen *I(k)* y se genera una serie de cajas delimitadoras para los posibles objetos frontales *M(k)*. *Z(k)* es la colección estimada de pixeles que pertenecen al plano frontal, y se emplea con *I(k)* para actualizar el modelo de fondo probabilístico a razón de tiempo de *k+1*.

a. **Cuantizacion:** Se guarda un histograma *Hij(k)* con un espacio de color RGB para cada pixel. Para *Hij(k)* se realiza una suposición de tal forma que el número de colores exhibidos este limitado por un parámetro del sistema *Fmax*. Sin embargo la electrónica de una cámara genera un ruido que puede llegar a afectar a esta suposición, para mitigar este efecto se cuantiza este espacio de color mediante una cuantización lineal dado u parámetro $q < 256$.
b. **Inicialización del histograma:** Se emplean los primeros *T* fotogramas del video como datos de entrenamiento para inicializar el modelo del fondo. Se interpreta la probabilidad de *Hij(k)* como un vector R_{q^3} donde cada eje representa un único color. Se genera un *F* que representa el conjunto de todos los vectores unitarios en R_{q^3}. Se considera $f_{ij}\ (k)\ \in F$ como un rasgo (color del pixel) observado en un momento *k*. De todas las *T* formas observadas se selecciona las formas únicas más recientes cumpliendo con $F_{tot} \leq F_{max}$. Si se da el caso de que *T* sea mayor que F_{max} es posible que F_{tot} exceda el valor de F_{max} en ese caso se rechazan las observaciones más antiguas hasta cumplir la condición. Después se calcula la media para generar el histograma inicial y se le pone a F_{tot} un peso igual a $1/F_{tot}$.
c. **Inferencia Bayesiana:** Se emplea la regla de Bayes para calcular la probabilidad de que un pixel sea clasificado como perteneciente al fondo o al frente dado un rasgo $f_{ij}\ (k)$. Empleando la regla de Bayes y la regla de la probabilidad total se obtiene:

$$p(B|f) = \frac{p(f|B)p(B)}{p(f|B)p(B) + p(f|F)p(F)}$$

Se calcula el valor de $p(f|B) = f_{ij}(k)^T \hat{H}_{ij}(k)$ dado que $H_{ij}(k)$representa el modelo del fondo. Sin un modelo estadístico del primer plano no se puede calcular la regla de bayes explícitamente, así que es necesario hacer otra suposición en la que $p(f|F) = 1 - p(f|B)$, que tiene la gran propiedad de que si el resultado es igual a 1 entonces el pixel sin duda pertenece al fondo, y si es 0 entonces pertenece al plano frontal. Tras calcular la probabilidad de cada pixel se obtiene la imagen posterior *P(k)*.

d. **Filtrado y conexión de componentes:** Dada la imagen posterior *P(k)* se realizan varias operaciones de filtrado con el fin de preparar una imagen binaria para usar como entrada para el algoritmo de conexión de componentes. Se realiza una apertura morfológica que reducirá la probabilidad estimada de los pixeles que no están rodeados por una región de pixeles de alta probabilidad, suavizando así las anomalías. Seguidamente se realiza una operación de cerrado morfológico que aumenta la probabilidad de los pixeles que están cerca de las regiones de pixeles de alta probabilidad. Los dos pixeles en conjunto forman una especia de operación de suavizado dando lugar a una imagen de probabilidad modificada *^P(k)*.
   Ahora se aplica un umbral γ entre (0,1) sobre *^P(k)* para generar una imagen binaria *B(k)*. El umbral actúa como una regla de decisión, si un pixel de la imagen modificada es mayor o igual que el umbral, entonces en la imagen binaria valdrá 1, y 0 en el caso contrario. En este caso 1 corresponderá al plano frontal, y 0 al fondo.
   Después vuelven a aplicarse operaciones de apertura y cierre morfológico en esta ocasión sobre la imagen binaria con el objetivo delimitar todas las regiones del frente que sean menores que el kernel circular establecido como parámetro de diseño y para conectar regiones adyacentes, mediante el proceso de rellenar aquellas regiones demasiado pequeñas para encajar con dicho kernel si no se superponen a otras existentes.
   En la imagen resultante con el algoritmo de los componentes conectados se detectan aquellas regiones de pixeles etiquetadas como pertenecientes al primer plano y mediante una función de OpenCV que se detallará en la sección de diseño, se obtienen las cajas delimitantes de dichos componentes y estas serán los valores de entrada para el sistema de seguimiento.
e. **Actualizar el histograma:** El algoritmo de seguimiento recibe como entrada *M(k)*, la lista de objetos identificados como del frente, y devuelve *Z(k)* el conjunto de pixeles identificados como del primer plano. Para actualizar el histograma se hace uso de la función $f_{ij}(k)$.

De esta forma se obtienen los objetos que GMG considera que no pertenecen al modelo del fondo.

Background substraction es un mismo enfoque, sin embargo se ha decidido implementar dos algoritmos diferentes ya que aunque ambos se basan en el mismo principio, separar el foreground del background, ambos lo hacen de formas muy distintas, cada uno tiene su manera de lidiar con los cambios en la iluminación, crear el modelo, etc… Se quiere ver cual funciona mejor en este dominio.

Implementación

MOG2: Ya se ha explicado cómo funcionaba este algoritmo de background substraction. Ahora hay que hacerlo funcionar para el sistema que se ha diseñado.

OpenCV implementa la mayor parte del algoritmo, pero aun así se necesita realizar varios ajustes para que funcione para el dominio de este problema. Lo primero es crear el algoritmo mediante su función correspondiente, se deben declarar tres variables para el mismo. El histórico, este número determina cuantos frames se usan para construir el modelo del fondo, básicamente si un objeto se mantiene quieto en una posición tantos frames como el marcado en el histórico se convertirá en background. Los murciélagos son objetos que se mueven mucho y muy deprisa, prácticamente cada frame, por lo tanto nos interesa que este número sea bajo ya que prácticamente nada se mueve tanto como los murciélagos, se debe tener cuidado con los murciélagos que se mueven de frente a la cámara, pues da la impresión de que no se mueven. 10 es el valor que se ha establecido. Sigue el valor umbral, el valor que determina como de bien descrito está un pixel con respecto al modelo, cuanto más alto es el valor más tiene que cambiar un pixel entre frames para ser contado como foreground. Por último el algoritmo de MOG2 puede detectar las sombras y señalarlas, hacerlo ralentiza ligeramente la velocidad del modelo pero ayuda a distinguir mejor los objetos de sus sombras, se decide activarla.

El modelo ya está, y en un bucle se aplica sobre cada frame uno a uno, pero esto solo no es suficiente, el algoritmo no es perfecto y su salida suele contener bastante ruido y huecos que hay que filtrar. Para ello sobre la máscara que genera el modelo se aplica una dilatación, esto ayuda a cerrar los posibles huecos que haya dentro de un mismo objeto y mantenerlo como un ente conjunto.

Figura 24: ejemplo MOG2

De esta manera se puede aplicar una función de extracción de contornos que devuelve los contornos de todos los conjuntos en la imagen generada. Sobre estos contornos se dibuja una caja delimitadora y esta se le pasa al tracker.

GMG: Por su parte, para emplear gmg solo se debe establecer en que frame empezará. Ya se comentó en la explicación del mismo que gmg emplea un número concreto de frames para generar un modelo, y no ese estabiliza hasta pasados esos frames. Este valor se ha decidido a base de pruebas, no siempre cuantos más frames mejo el modelo, se debe pensar que cuantos más frames más ruido captará el modelo también, pero un número muy bajo hace que gmg no genere un buen modelo, 100 son los frames que se han probado eficaces. Al igual que con el anterior se debe declarar el valor umbral, la única diferencia es que en este caso el valor es un número comprendido entre 0 y 1, un porcentaje, otra vez cuanto más cercano a 1 más exigente es el modelo a la hora de considerar un objeto como cambiante. También a base de prueba y error se ha establecido un valor de decisión del 0.9.

Después de aplicar el modelo sobre la imagen, la salida generada requiere alguna transformación más que MOG2. Concretamente dos que se comentaron en la explicación teórica, apertura y cierre morfológico, estas dos operaciones ayudan al igual que la operación de dilatación en el MOG2 a cerrar las zonas y en general a crear conjuntos más definidos.

Y se finaliza extrayendo los contornos de estos conjuntos, generando la caja delimitadora para cada uno de ellos y pasándoselo al tracker.

4.3 Función de Seguimiento

Como ya se comentó cuando se explicó el problema y se repitió en el apartado anterior de los detectores, en el problema a resolver, los objetos que se desean seguir son muchos y constantemente están saliendo del plano y entrando, algunos son nuevos y otros no. Por esto es que el detector debe aplicarse sobre cada uno de los frames del video, debe estar activado todo el tiempo por si entran nuevos objetos. Se dice esto porque una aproximación habitual es hacer que el detector actué una vez cada varios frames, y entre medias dejárselo al algoritmo de seguimiento ya que por lo general estos son más ligeros, y de esta forma se ahorra procesamiento. Pero en este caso no vale esta aproximación.

Puesto que el detector está funcionando constantemente el algoritmo de seguimiento debe centrarse solo en las tareas que el detector no puede. Lo que se busca del tracker es que cumpla dos tareas, que sea capaz de mantener en lo posible la identidad de un individuo a través de los fotogramas, y que si el detector lo perdiese, tenga una manera de seguir identificándolo.

Se probaron varios algoritmos con distinto funcionamiento. Se implementó un algoritmo de Kalman que basa su seguimiento en predicciones, según la trayectoria y velocidad del objeto calcula donde va a estar en el siguiente fotograma. Los resultados no fueron buenos, el vuelo de los murciélagos es demasiado errático y las predicciones nunca

acertaban. Se probó con algoritmos más potentes que están pensados para seguimiento de un solo objeto como kcf o csrt, tenían dos problemas. Son algoritmos muy potentes que casi nunca pierden un objeto, no es este el caso ya que siempre perdían en algún momento el objetivo, generalmente porque entraba en una zona de sombras y se hacía prácticamente indistinguible y estos algoritmos no volvían a relacionarlo al salir de la sombra. El otro problema es que dado que están diseñados para seguir a un solo objeto, cada vez que se le pasa uno nuevo, la capacidad del procesador se divide, y o se cuenta con un procesador muy potente, que no es el caso en el equipo de este proyecto, o se emplea procesamiento mediante GPU para lidiar con el trabajo, pues de lo contrario el programa se vuelve extremadamente lento. En este proyecto el ordenador empelado no tenía la capacidad de usar GPU, y sin eso los tiempos eran de un fotograma cada varios segundos, algo inaceptable.

Finalmente se escogió un algoritmo muy ligero, uno que pertenece a la categoría de los basados en puntos.

Algoritmo

Asignación de ID basado en centroides: Es una aproximación que se basa su funcionamiento en las distancias entre los centros, o centroides de un objeto existente y los centros de nuevos objetos que vayan apareciendo.

Este tipo de algoritmos son multi-proceso y su funcionamiento puede resumirse en 3 pasos.

1. El algoritmo requiere recibir como entrada una caja delimitadora que marque la posición del objeto que se desee seguir (en este caso estas coordenadas las da el sistema de detección). El método calcula el centro de coordenadas de la caja y le asigna un identificador id.

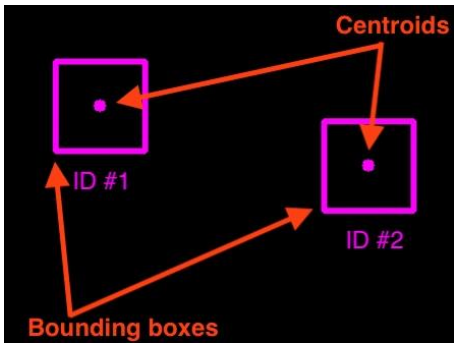


Figura 25: centroide 1

2. Para cada nuevo fotograma del video se volverán a buscar los centroides como en el paso anterior sin embargo en lugar de asignar nuevos identificadores a cada una de las detecciones se tratará de asociar el nuevo centroide con el antiguo, el del fotograma anterior. Para ello se calcula la distancia (Euclidea, Gaussiana, etc…)

entre los antiguos y los nuevos para intentar asociarlos, con la suposición de que a menor distancia, más probable es que sean el mismo objeto.

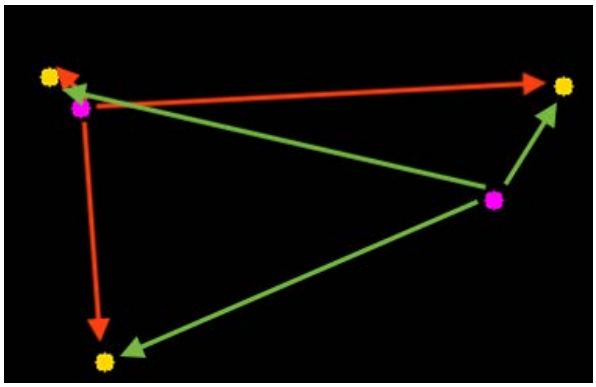


Figura 26:centroide 2

3. Pero siempre aparecerán nuevos centros, o será imposible relacionar dos instancias. Cuando se da el caso de no poder relacionar un centroide nuevo con uno antiguo, este se registra como uno nuevo.
   Del mismo modo si un centroide desaparece, ya sea porque el detector lo ha perdido o se ha salido de la pantalla, este se elimina del registro.

Figura 27: centroide 3

Figura 28: centroide 4

Implementación

Para la implementación de este sistema de centroides se crearon dos clases aparte, una que gestiona la asignación, eliminación y asociación de los centroides y otra clase que contiene el objeto objetoTrackeable en la que se registra un historial de las posiciones en las que ha estado el centroide con el id X, y si ese id ha sido o no contado, aunque este último valor se explicará en la función de conteo. La clase del centroide se compone de cuatro métodos, el inicial donde se inicializan las variables a emplear, el de registro que registra nuevos centroides, el de desregistro que elimina los id no asociados a un centroide, y el método de actualización, el más importante que gestiona a los dos anteriores y es al que se llama desde el programa principal.

El funcionamiento de la clase es simple, se inician las variables de siguiente id de objeto un número que indica cual es el siguiente id a asignar, dos diccionarios ordenados en los que se almacenan las objetos que se están siguiendo actualmente y en el otro los objetos que han desaparecido, y por último una variable que indica cuanto tiempo se desea que se mantengan los objetos en el diccionario de desaparecidos antes de eliminarlos completamente, de esta forma si un objeto desaparece en unos pocos fotogramas, al aparecer tiene todas las probabilidades de ser asignado al mismo id que tenía porque seguirá estando muy cerca.

Cuando el detector obtiene las coordenadas de las cajas delimitadoras de un frame, las guarda en un array y se lo pasa al método de actualización del tracker. Si el array está vacío directamente se añade + 1 a la cuenta de los id en desaparecidos y si alguno excede el máximo que se ha determinado se elimina. Si no está vacío se calculan los centroides a partir de las coordenadas de las cajas y se guardan los centroides, si no se está siguiendo ningún objeto en ese momento, se registran directamente estos nuevos centroides, si ya se estaba siguiendo alguno (la mayoría de los casos) hay que asociar estos centroides con los ya existentes

Para asociarlos se hace uso de la distancia euclídea, la función es un poco larga e implementa varios bucles y comprobaciones, así que se explicara de forma más genérica que paso a paso ya que son muchos. Por cada objeto con su centroide asociado que se esté siguiendo se calcula la distancia euclídea a cada uno de los centroides recién calculados y se les asigna el más cercano en orden. Si el número de objetos registrados es mayor que el número de centroides actuales, se comprueba que hayan desaparecido y se registran como tal o se eliminan si el objeto ya estaba marcado como desaparecido y ha superado el tiempo de espera. Si se da el caso de que los centroides que han llegado son más que los objetos registrados, se deben registrar los nuevos centroides que queden sin asociar como objetos trakeables nuevos.

La clase devuelve la lista de objetos con su id y centroide asociado de vuelta al programa principal donde se pintan en la imagen del fotograma y se guardan en una variable que empleará la función de conteo.

4.4 Función de Conteo

Esta función es la que se tiene que encargar de llevar la cuenta de los murciélagos en pantalla lo más acertadamente posible. El diseño de la misma está basado en la forma en la que contaría los murciélagos del video una persona.

Cada video tiene una distribución distinta, pero todos tienen dos cosas en común, un punto de entrada y salida de los murciélagos a la colonia y los límites del video. Pesto que la función tiene que funcionar en todos los videos, se debe diseñar de forma que aproveche las características comunes de estos.

Cuando se cuentan los murciélagos a mano (con el video ralentizado evidentemente) una persona tiene fácil identificar cuáles son cada uno de los murciélagos, una persona es el detector y trakeador ideal, de modo que lo único que no sabe la persona que los cuenta es lo que esta fuera de los límites del video, el interior de la colonia y más allá de los bordes, durante el resto del tiempo se conoce el estado del murciélago. Resumiendo, para un conteo efectivo se debe contar al murciélago o bien cuando aparece en la pantalla o cuando desaparece de la misma por una de estas zonas límites.

Por ilustrarlo, figura 29:

Figura 29: video 1 Entorno

Esta imagen, figura 29, pertenece al video 1. En ella se ve la entrada a la cueva y un poco de los alrededores. Como la entrada de la cueva suele estar lejos y a menudo salen varios murciélagos a la vez, solapándose en la imagen los unos con los otros, se ha decidido que la función cuente a los murciélagos cuando estos desaparecen de la imagen en vez de cuando aparecen. Como hacerlo:

Antes de empezar a reproducir el vídeo se pinta mediante una función manual con el ratón, una región de interés. El tamaño y posicionamiento de esta región se establece tras haber mirado el video de antemano, para saber por dónde y cómo salen los murciélagos, figura 30.

Figura 30: zona de interés

El conteo en esta región funciona de esta manera. Cualquier centroide asignado a un objeto que este dentro de la zona roja, figura 30, cuya posición anterior este fuera de la zona y cuyo desplazamiento sea considerable como tal, 20 pixeles en este caso. El objeto que cumpla esta condición se considera que está entrando en la cueva, y por lo tanto se añade +1 a la cuenta de murciélagos y este objeto se etiqueta como contado, para evitar que si no entra realmente y vuelve a salir, cuando entre de nuevo no sea contado otra vez, figura 31.

Figura 31: centroide desplazamiento

De forma similar, en el caso de que el murciélago se salga por los límites de la pantalla, se contará cuando su posición actual este dentro de la zona borde, y su posición anterior fuera de esta, con un desplazamiento entre ambas considerado como tal. Esta región de los bordes no se escoge manualmente, esta implementada con unas dimensiones fijas, que se obtienen de restar a las dimensiones del video un margen de 180 pixeles al ancho y 60 a la altura. Este número se ha elegido tras varias observaciones y pruebas.

Resumiéndolo tal como está en la implementación, se pinta el área de conteo, si la posición de un centroide en el fotograma actual está dentro de esa zona, se comprueba en su registro la posición anterior si la tiene (si no es nuevo a sí que se continúa como si nada) si esta está fuera de la región límite se mira la diferencia de posición entre ambas para el eje x y el y por separado, si el valor absoluto de esta diferencia es superior a 20 pixeles en el eje x se mira el signo del resultado de la diferencia, negativo se etiqueta como desplazamiento hacia la derecha, positivo hacia la izquierda. Lo mismo para el eje y, negativo hacia abajo, positivo hacia arriba.

El motivo de estas etiquetas es que por defecto se cuentan los objetos entren por donde entren, pero dependiendo del video, se puede desear que solo se cuenten si por ejemplo vienen por la derecha, esto se declara en como argumento de entrada al ejecutar el video. De esta forma si el argumento está vacío se marcaran como contados y se aumentara la cifra de conteo para cualquier objeto que entre en la zona, si se establece una dirección como por ejemple derecha, solo se contarán los objetos que tengan esa etiqueta.

La función para los bordes límites es igual pero más simple.

Y de esta forma se cuentan los murciélagos. Lo bueno de esta función es que ignora gran parte del ruido de los pasos anteriores con ese filtro de considerar algo como desplazamiento, ya que rara es la ocasión en que una rama se mueve tanto como para considerarse desplazamiento. Lo malo de esta función es que no está cubierta a fallos por parte del detector y el traker, por ejemplo si dos murciélagos pasan tan cerca que sus id se intercambian, y uno estaba etiquetado como contado y el otro no, puede dar lugar a que uno de ellos no se cuente o a que otro se cuente dos veces. Pero esto se estudiará más en las pruebas.

Y estas son las cuatro soluciones diseñadas, cuatro detectores diferentes combinados con un traker y la función de conteo. A continuación se verá juzgará su rendimiento mediante distintas pruebas.

5. PRUEBAS Y RESULTADOS EXPERIMENTALES

5.1 Base de datos empleada

Para la realización de las pruebas experimentales de este proyecto se ha empleado un total de 4 videos proporcionados por el equipo de control de plagas… que muestran distintas zonas de aparición de poblaciones de murciélagos en la península, cada uno con distinta iluminación, composición, distancia al objetivo, elementos irruptores, etc...

Todos los videos comparten la característica de ser en cámara nocturna y grabar la entrada y salida de una colonia.

Son grabaciones con dimensiones dispares, así que en los códigos han sido redimensionados a 854 pixeles de ancho y 480 de alto, que son las dimensiones del primer video.

El primer video tiene la característica de mostrar los vuelos más erráticos de todos, puesto que la grabación está realizada a una distancia considerable y la entrada es bastante ancha los murciélagos se mueven de forma muy poco predecible. Otra característica es que tiene bastante ruido en la periferia de la pantalla, hay muchos árboles y ramas que dificultan y confunden al algoritmo. Contiene tamaños variables de murciélagos, la mayoría están en la lejanía, pero otros pasan muy cerca de la cámara. Es el video más completo de los 4.

El segundo es el más largo de los cuatro, con más de 20 minutos de duración, está realizado sobre una entrada muy grande y a bastante distancia, a diferencia del video anterior, la cámara no está en la trayectoria de salida de los murciélagos, por lo que la gran mayoría de ellos pasaran lejos de la cama y se verán muy pequeños. Otra característica única de este video es que no tiene casi nada de ruido proveniente de los objetos fijos en el mismo, apenas hay unas pocas ramas de una planta pequeña a la derecha de la pantalla, sin embargo es el video que más partículas irruptoras posee. Con bastante frecuencia aparecen en la imagen lo que parecen ser hojas caídas u objetos similares que manchan la detección.

Los videos tercero y cuarto comparten el mismo fondo, son grabaciones realizadas sobre la misma entrada y en la misma posición, la diferencia entre ambas es que en la cuarta los murciélagos salen en la mayoría de los casos uno a uno y bastante espaciados entre sí, hay poca afluencia, en cambio en el tercero salen muchos más murciélagos aunque muchos de ellos se mantienen en la ventana de salida sin llegar a salir del todo. Estos dos videos son con diferencia los que más ruido ambiental tienen, la entrada esta junto a una gran cantidad de matorrales y otras plantas, y la cámara está situada a baja altura y capta la hierba del suelo.

5.2 Descripción de las pruebas experimentales

En esta sección se van a realizar varias pruebas de carácter experimental con el objetivo de comprobar la eficacia y precisión de los distintos sistemas implementados. Las pruebas se realizarán sobre los videos mencionados anteriormente.

Las pruebas consistirán primero en una comprobación visual de su funcionamiento, escogiendo cada video como entrada y examinando los puntos que el sistema capte comprobando si cumple mínimamente con la tarea de detectar y seguir a los objetos deseados.

Además se realizara un estudio de tipo cuantitativo comparativo con el objetivo de medir el rendimiento de los distintos sistemas entre sí. Con esto se quiere obtener que sistema de los implementados realiza la tarea con mayor eficacia e ilustrar cual es el más adecuado para la tarea.

5.2.1 Análisis cualitativo

En esta sección se realizará el análisis cualitativo de los datos, que como ya se ha mencionado consistirá analizar todos los videos con cada uno de los sistemas implementados y comprobar si el sistema detecta y sigue los objetos de interés, si se equivoca a menudo, como maneja objetos lejanos, oclusión, colisión y otros aspectos similares.

Los sistemas que se analizarán son, las cuatro implementaciones: las dos orientadas al reconocimiento, ORB y Haar-cascade, y los dos algoritmos de background. Los cuatro apoyándose en el sistema de tracking basado en centroides.

En el caso de los sistemas de background, en cada uno se examinarán no solo la salida de video con las cajas delimitadoras y los centros de las mismas, sino que también se examinaran los videos de las máscaras con los umbrales que ilustran muy bien el ruido que captan cada uno de los sistemas y la nitidez con la que detectan los objetos de interés. En las imágenes de las cajas no se deben confundir los elementos que actualmente se están siguiendo, con los centroides que no se han asociado con ningún objeto porque este ha desaparecido y están a la espera en unos pocos fotogramas de que aparezca un objeto con el que sea plausible relacionarles. Estos se distinguen porque no tienen ninguna caja delimitadora alrededor solo son un punto y su id.

ORB

Este acercamiento ha resultado ser muy poco adecuado para los videos que se tratan en este documento, este sistema de detección tiene dos problemas papables, el primero es lo poco flexible que resulta su sistema. Emplea una fotografía como referencia para buscar otras similares, pero los murciélagos están constantemente moviéndose, girando batiendo las alas cerrándolas, en definitiva están cambiando de forma constantemente, esto provoca que la gran mayoría de murciélagos no los capte si no están en posiciones

similares a su modelo de referencia, especialmente entre videos con diferencias en la iluminación. El segundo problema es el propio dominio en sí. La calidad de las grabaciones es muy pobre, los murciélagos se mueven muy deprisa y son muy pequeños con lo que en la práctica totalidad de los fotogramas estos se ven borrosos y distorsionados, no tienen esquinas u otros rasgos suficientes como para ser identificables. En muy pocas ocasiones es capaz de detectar algún murciélago, y cuando lo hace nunca lo aísla de objetos similares.

En las imágenes a continuación, figuras 32 y 33, se ilustra muy bien el problema que sufre este modelo en las ocasiones en que es capaz de detectar algún murciélago.

Figura 32: ORB prueba 1

Figura 33: ORB prueba 2

La caja delimitadora de la figura 33 se logra generar cuando se aplica sobre una imagen, ya que cuando se hace funcionar sobre un video, los resultados que marca generan conflictos con el tracker y como resultado el sistema falla.

Estas pruebas, aunque en primera instancia muestran que el acercamiento por coincidencia de formas no es apto para resolver el problema, lo cierto es que arroja mucha información útil acerca del problema de los murciélagos.

En el resto de videos tiene los mismos problemas, más acentuados aún en el 3 y 4.

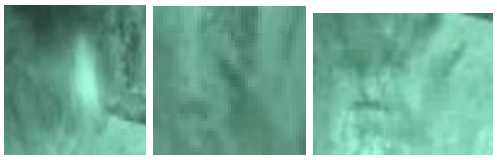
Haar-Cascade

El segundo acercamiento hacia el reconocimiento del objeto y posterior localización, y el único que emplea técnicas de aprendizaje.

Lo cierto es que aunque es más potente que el sistema de coincidencia de formas y es mucho más adaptable a las variaciones, sufre de problemas muy similares. Para entrenar este sistema en la detección de los murciélagos, se emplearon capturas de todo tipo, tanto de murciélagos lejanos como cercanos, más nítidos y menos, en sombras, en zonas iluminadas, etc… se intentó cubrir todos los modelos, ya que en necesario que localice a los murciélagos estén donde estén.

Algunas muestras son:


Video 1


Video 2


Video 3 y 4

En el video 1 se ve muy bien cuál es el principal problema de esta aproximación a este dominio.

Figura 34: Haar prueba 1

En esta primera ilustración, figura 34, se quiere resaltar los dos id más arriba de la imagen, el id 6 y el id 5. Estas dos marcas son recurrentes a lo largo de toda la detección, el sistema prácticamente las marca en todos los frames, el motivo es muy simple, si se miran de cerca, uno pertenece a una roca en la pared más clara que su alrededor y la otra es la hoja de una rama que esta sobre un fondo oscuro. Nada las distingue de las imágenes de los murciélagos que se le han dado como muestra para reconocer, y lo mismo ocurre en varias ocasiones con otras hojas y marcas en las paredes de la imagen.

Figura 35: Haar prueba 2

Esta segunda imagen, figura 35, muestra cuando marca a los murciélagos, cuando están cerca de la cámara y sus formas están más definidas, si no están como mínimo así de cerca, el sistema no los marca regularmente. El sistema parece que solo marca puntos claros en zonas oscuras, y murciélagos muy de cerca, y nunca durante más de tres o cuatro frames. No parece muy alentador.

En el video 2, es más de lo mismo, los murciélagos son escasos y cuando aparecen a no ser que estén pasando por una zona oscura o de cerca (Cosa que en este video no pasa casi nunca) el sistema no los marca. Pero vuelve a marcar constantemente las hojas solitarias en fondos más oscuros, figura 36.

Figura 36: Haar prueba 3

Los videos 3 y 4 son los que mas confirman esta tendencia que se ha estado viendo en los videos anteriores, en esta ocasión es mas positivo. Aunque de vez en cuando marca alguna hoja que otra, lo que marca mas amenudo son los murcielagos dentro de la ventana, claro sobre oscuro. Por lo que se ve, figuras 37 y 38, solo marca a los murcielagos cuando o bien la forma esta mas definida, o cuando el contraste entre ellos y el fondo es muy notorio. Y esto segundo es lo que provoca que se marquen otras hojas y ramas.

Figura 37: Haar prueba 4

Figura 38: Haar prueba 5

Los motivos son bastante claros, en el segundo video los murciélagos son demasiado pequeños y están demasiado lejos en todo momento por eso los capta muy raramente, en el video 1 solo detecta a los murciélagos más cercanos a la pantalla y siempre cuando las

alas son reconocibles. En los videos 3 y 4 las formas de los murciélagos son algo más nítidas, pero solo dentro de la ventana, fuera nunca los capta, el contraste tiene que ser muy marcado para que los reconozca como murciélagos.

Y es que ese es el mayor problema que se enfrenta en estos videos y que afrontan los métodos de reconocimiento directamente. Las formas de los murciélagos no se distinguen. Ya las técnicas más modernas y potentes de aprendizaje tienen problemas para tratar con los objetos pequeños y las técnicas del acercamiento clásico nunca tuvieron demasiada precisión incluso cuando detectaban objetos nítidos. Aquí los murciélagos se ven como puntos más claros o borrones, que no se diferencian en casi nada de otros elementos de la imagen, solo la cercanía y el contraste marcado dan resultados.

Como ya se explicó en su sección correspondiente basta con que uno de los clasificadores de la cascada marque un objeto como no objeto, para que sea irrecuperable. La fuerza de esos clasificadores proviene de la consecución, que todos los clasificadores acepten una entrada hace que tenga grandes probabilidades de serlo.

Las imágenes de los murciélagos que se han visto están ampliadas y enfocadas para ver a los murciélagos, pero si se alejan un poco nada los distingue de marcas en la pared o de sombras. Es muy fácil que sean rechazados por uno de los clasificadores y que por tanto ninguna de las entradas llegue al final.

Estas pruebas ya establecen que al acercamiento del reconocimiento mediante técnicas de cascada, le falta potencia de aprendizaje para poder discernir que no todos los puntos luminosos son murciélagos, y su método de clasificación basado en las intensidades sufre con los murciélagos en zonas más iluminadas.

Background MOG2

Este detector ha demostrado unos resultados visuales muy buenos. Como ya se ha explicado en secciones pasadas, para este algoritmo se estableció un umbral de detección relativamente alto, con la intención de eliminar ruidos pequeños pero permitiendo que detectase los cambios de los murciélagos más lejanos en el plano. También se estableció que se distinguiesen las sombras, lo que permite evitar los ruidos de estas en videos como el 1 y además permite captar mejor partes borrosas de un murciélago como podrían ser las alas en movimiento. Se recuerda que las sombras o partes que cambian más levemente, son representadas con un color gris en la imagen de la máscara, los cambios más intensos se marcan como blancas.

Video 1

Figura 39: MOG2 prueba 1

En esta primera imagen, figura 39, hay tres murciélagos que han sido señalados, aunque dos de ellos están cercanos en el plano aunque lejanos en distancia real. Uno de los murciélagos está muy atrás en el plano, entrando en el túnel, momento en el que otro ha pasado más cerca de la cámara por la misma zona, esto ha resultado en dos posibles marcas y el sistema ha separado un ala del murciélago más al frente considerándola con el de detrás. Aunque hay varios puntos en la imagen normal, varios de ellos pertenecen a centroides sueltos, que no están asociados a ningún objeto. En la imagen de la máscara se ve más claro que hay dos secciones diferenciadas. En definitiva los consigue marcar más o menos pero hay mucha confusión con la asignación de los id ya que están casi solapándose. También ha detectado un ligero ruido, la pinta de una rama, que ha señalado.

Figura 40: MOG2 prueba 1 mask

En esta segunda imagen, figura 40, ya no hay solapamiento de objetos, están mucho más separados y pertenecen más o menos al mismo plano y como tal se puede ver que el sistema apenas detecta nada de ruido. En este fotograma no hay ningún reconocimiento

que no sea un murciélago, y se señala la posición de los mismos de forma individual y correcta. En el video en general esporádicamente señala alguna rama de los árboles, la punta concretamente que es la que más se mueve, pero son pocas las ocasiones y es despreciable, nada que influya realmente en el recuento. Ejemplos de murciélagos más lejanos, figuras 41 y 42.

Figura 41: MOG2 prueba 2

Figura 42: MOG2 prueba 2 mask

Video 2

Este es el video más largo pero con menos ruido ambiente. Ya se ha visto en el video anterior que el sistema de MOG2 apenas detecta ruido objetos del fondo. En este caso las ocasiones en las que detecta algo, ocurre en los matorrales a la derecha de la imagen, pero son casos muy aislados, prácticamente solo señala a los murciélagos, lo que de hecho consigue con bastante precisión. Si es cierto que en este video aparecen partículas bastante notorias en la imagen (presumiblemente son hojas de árbol cayendo o insectos cercanos a la cámara como polillas) y el sistema las captura como si fuesen murciélagos,

ya que se mueven casi igual de rápido y son casi indistinguibles incluso a ojo humano. Pero la imagen inferior, figuras 43 y 44, ilustran muy bien lo que ocurre en casi la totalidad del video, señala exclusivamente a los murciélagos.

Figura 43: MOG2 prueba 3

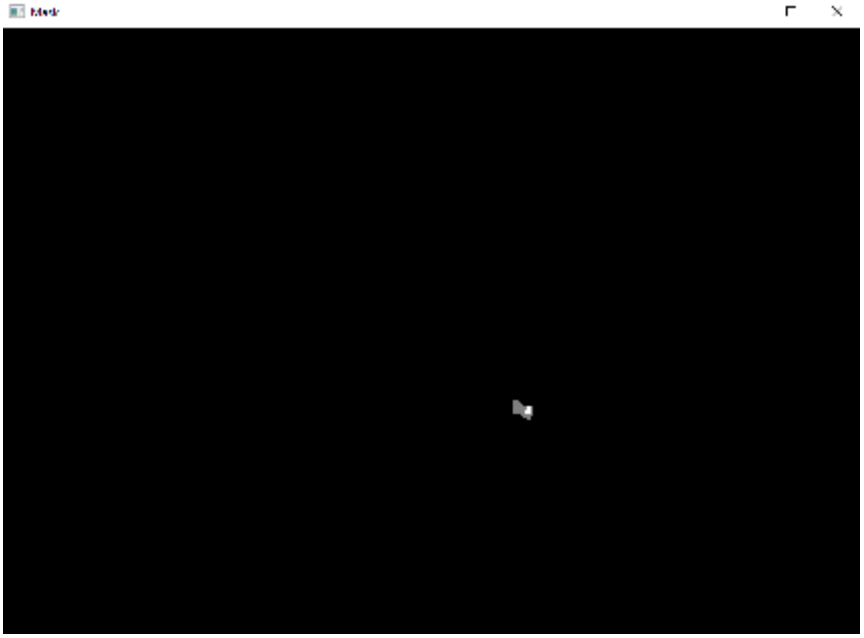


Figura 44: MOG2 prueba 3 mask

Videos 3 y 4

Se juntan los dos videos porque los resultados visuales son muy similares, la mayor diferencia como ya se ha dicho es la diferencia en el flujo de murciélagos entre uno y otro, pero eso será de interés y por tanto se medirá en la cuantitativa.

Figura 45: MOG2 prueba 4

En esta primera captura, figura 45, se observan dos cosas, la primera es que al contrario que en los videos anteriores el sistema ha captado más ruido en las ramas de la imagen, concretamente la detección de movimiento en las espigas de la parte inferior derecha será bastante habitual en todo el video. Esto es debido a la cercanía en este video, a diferencia de los anteriores, la grabación se ha realizado desde mucho más cerca de la salida de murciélagos y esto tiene varias implicaciones. La primera es que se pueden detectar a los murciélagos moviéndose dentro de la salida sin que hayan llegado a salir realmente, lo que trae problemas a la hora de contarlos, pero como se muestra en la imagen inferior, figura 46, no ocurre así con señalarlos, se les detecta bien.

Figura 46: MOG2 prueba 4 mask

Las implicaciones más graves de realizar la grabación cerca de la salida se observan mejor en las siguientes imagenes, figura 47 y 48. Para realzar las grabaciones de estos videos, siempre se ilumina la zona de la salida de los murciélagos para verlos mejor, al menos con el ojo humano. En los videos grabados a más distancia no hay problemas aparentes, pero estos dos videos 3 y 4 son los que están grabados a una menor distancia, y por tanto no es solo la salida lo que está iluminado, sino la práctica totalidad de la imagen, esto provoca que los tonos de los murciélagos sean muy similares a los del fondo y que en ocasiones los cambios detectados en la intensidad de los pixeles por parte de los algoritmos de detección sean inconsistentes. En la imagen anterior se veía como se marcaba bien al murciélago en el interior del edificio, sin embargo en esta otra imagen también hay un solo murciélago pero la diferencia es que este, esta fuera y con las plantas como fondo. La imagen normal muestra multitud de pequeñas cajas señalando la posición del murciélago, distintas partes de él. En la imagen de la máscara se ve más fácilmente como en vez de formar un solo bloque, el murciélago está compuesto de pequeñas zonas fragmentadas que el algoritmo no ha podido rellenar de forma consistente, debido al poco contraste entre los pixeles. Un problema que afecta a la medición de los murciélagos que salen de la pantalla.

Figura 47: MOG2 prueba 5

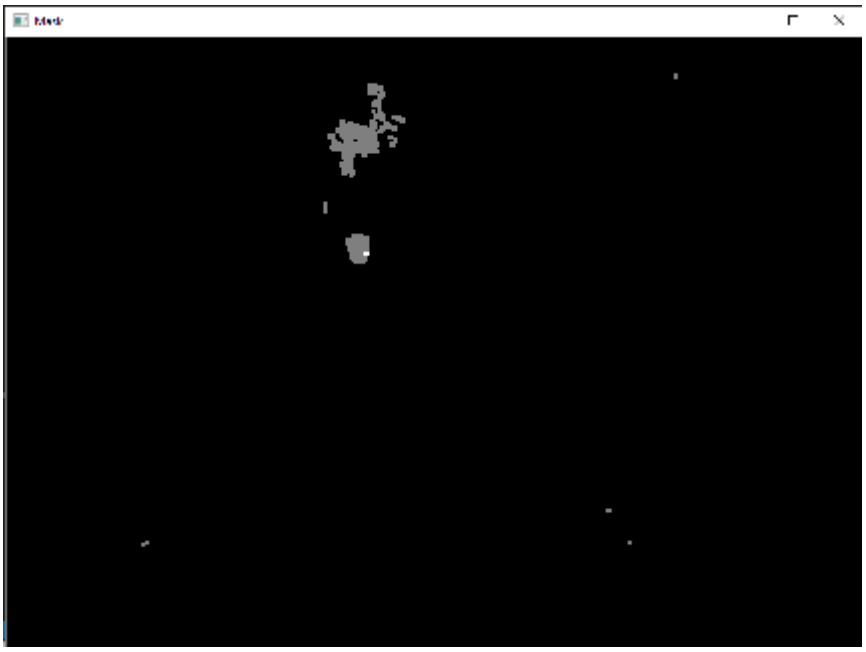


Figura 48: MOG2 prueba 5 mask

Background GMG

Esta combinación no ha mostrado los mejores resultados, aunque como se observa en las imágenes el sistema capta muy bien a los murciélagos, también hace lo propio con las ramas y partículas que pasan por la pantalla, en definitiva capta demasiado ruido. Esto es fácilmente justificable ya que como se mencionó en secciones pasadas, GMG necesita dejar pasar varios frames hasta empezar a realizar buenas detecciones, pues emplea esos primeros fotogramas para crear un modelo con el que analizar los siguientes. Esto genera dos problemas principalmente, el primero que en los primeros segundos de la reproducción del video el sistema capta prácticamente todo como movimiento, dando

lugar a muchas mediciones falsas y equivocadas que alteran las medidas. El segundo problema depende más del video en cuestión como se ilustra en las imágenes X, tómese el video primero, el V1, ya en las primeras instancias de video hay muchos murciélagos moviéndose por la pantalla, y el movimiento de las plantas, que aunque es más lento que el de los murciélagos, en el tiempo en el que tarda GMG en calibrarse, el movimiento de las plantas es mucho más notorio, esto provoca que GMG no tenga una secuencia de fotogramas "limpios" por así decirlo, de ruido con el que generar un modelo fiel al del fondo.

Video 1

Esta primera imagen, figura 49, corresponde con los primeros fotogramas del video 1, como se ha explicado estas primeras instancias son para modelar el fondo por lo tanto cualquier zona iluminada y mínimamente cambiante es marcada por el detector, evidentemente esto afecta notablemente a las mediciones finales que se obtienen como recuento, son datos llenos de falsos positivos.

Figura 49: GMG prueba 1

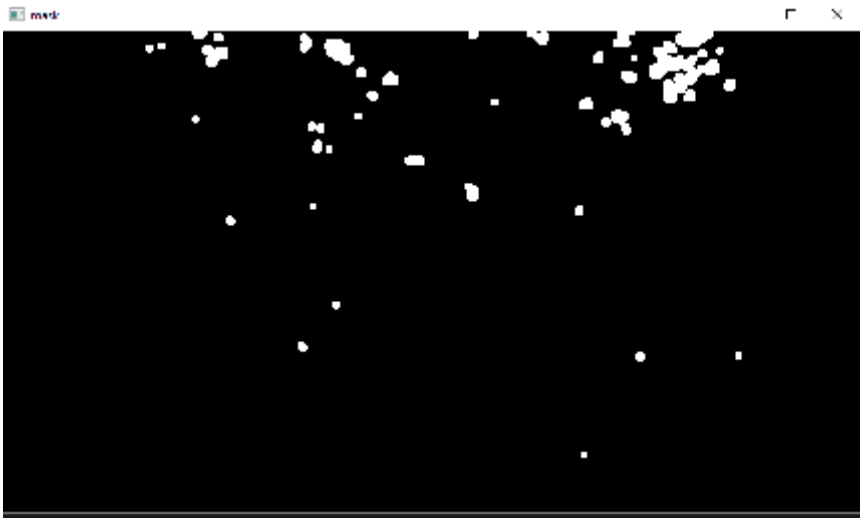


Figura 50: GMG prueba 1 mask

Cuando ya han pasado los primeros fotogramas y se ha construido el modelo, ya se puede ver en la imagen de la figura 51, que el programa capta muy bien a los murciélagos tanto de cerca como de lejos, y es importante remarcar que de cerca consigue señalar al murciélago como una sola entidad sin partirlo separando el ala del cuerpo por ejemplo. Sin embargo sigue cogiendo ruido de las ramas, y es normal, al fin y al cabo son elementos que se mueven, muy poco pero se mueven, a pesar de haber establecido un umbral de decisión muy alto para este algoritmo, se siguen captando los movimientos más rápidos en las puntas de las ramas.

Figura 51: GMG prueba 2

Figura 52: GMG prueba 2 mask

Video 2

Este segundo video es el mas largo de todos con mas de 20 minutos de grabación es el vide con menos ruido en el fondo, sin embargo es el video con mas particulas irrumpiendo

en la grabación. El poco ruido general se puede ver en la imagen de debajo, figuras 53 y 54, en el video anterior los primeros frames estaban marcados con multitud de puntos generados por el detector, en esta ocasión sin embargo hay muy poco (dentro de lo que podría ser).

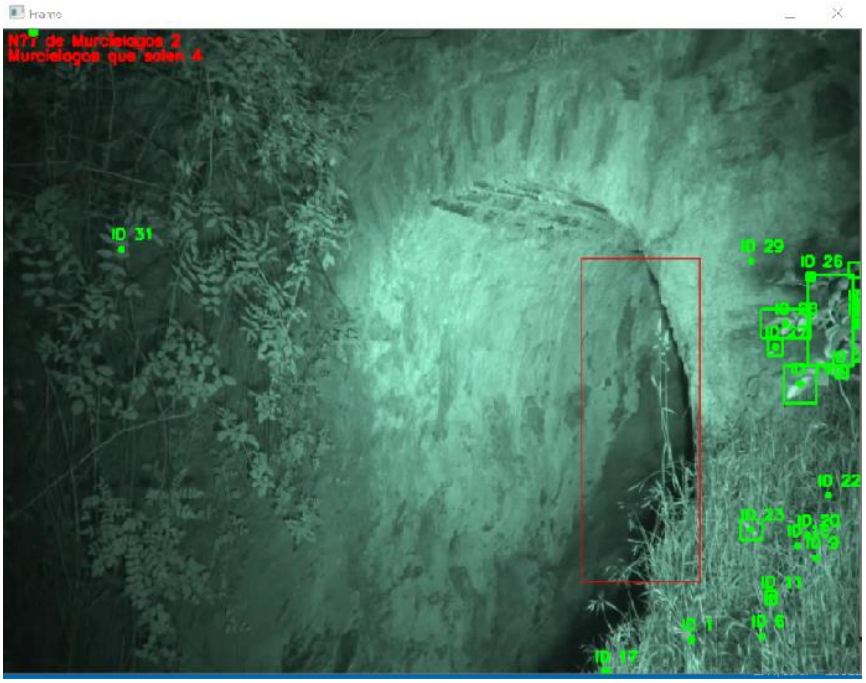


Figura 53: GMG prueba 3

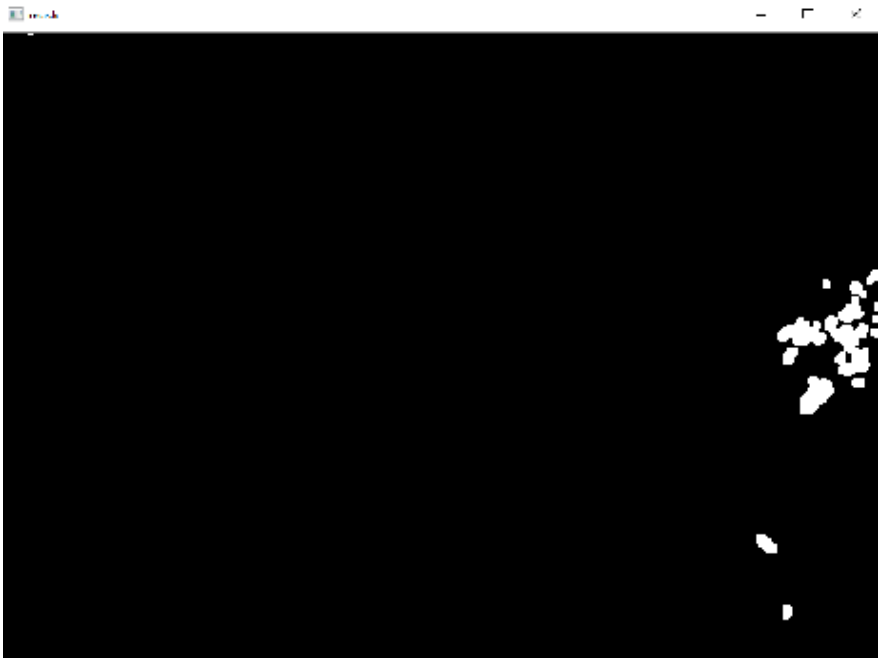


Figura 54: GMG prueba 3 mask

Ya regulado el modelo, el programa apenas capta ruido salvo las puntas de las ramas de la derecha que son las que más se mueven puntualmente, los murciélagos los capta a la perfección como se ve en ese punto solitario, en medio de la imagen, ese es el único murciélago del frame. Sin embargo se sigue observando que el algoritmo de GMG es

demasiado sensible, este video ilustra muy bien el problema de detectar estos animales solo por el movimiento (problema que se corrige con la función de conteo). Los murciélagos como el señalado en esta imagen, figura 55, son muy pequeños y prácticamente son puntos luminosos en la escena, en este video pasan bastantes partículas, seguramente hojas de algún árbol que son prácticamente indistinguibles de los murciélagos incluso ante el ojo humano y solo puede diferenciarse por el patrón de movimiento, cosa que ni el detector ni el tracker pueden distinguir, dando lugar a falsedades. Como ya se comentó esto se puede corregir con la ventana de conteo y la dirección de entrada deseada.

Figura 55: GMG prueba 4

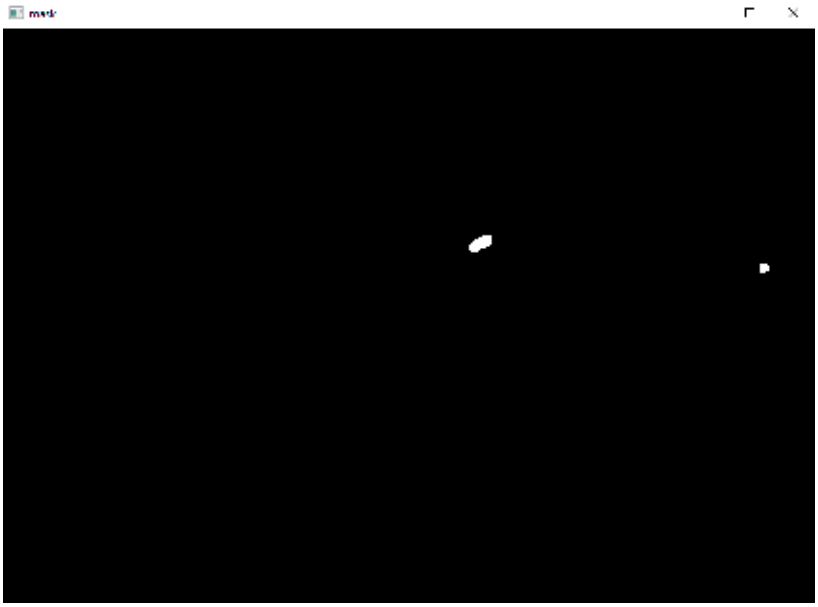


Figura 56: GMG prueba 4 mask

Video 3 y 4

Se juntan los dos videos porque los resultados visuales son muy similares, la mayor diferencia como ya se ha dicho es la diferencia en el flujo de murciélagos entre uno y otro, pero eso será de interés y por tanto se medirá en la cuantitativa.

Una vez más GMG capta una gran cantidad de ruido en sus primeros compases, y este video que es el que más ruido tiene no es una excepción. En la imagen, figura 57, dentro de la ventana se están moviendo dos murciélagos, sin embargo hay varias cajas delimitadoras, señalando más de un murciélago. En un principio puede pensarse que esto es así porque el fondo aún no está calibrado, pero no es el caso.

Figura 57: GMG prueba 5

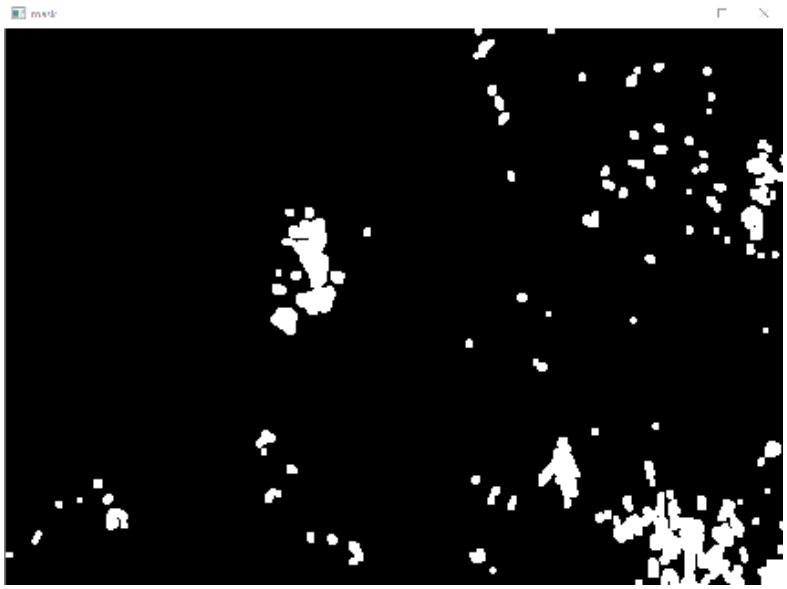


Figura 58: GMG prueba 5 mask

Una vez calibrado el fondo ya se puede ver que no capta casi ruido a pesar de que muchos elementos lo pueden provocar, las espigas que se muestran en la esquina inferior derecha de la imagen son constantes a lo largo de todo el video, no se mueven demasiado, pero al estar tan cerca su movimiento se remarca.

Sin embargo el ruido ambiente ya se ha observado suficiente en los videos anteriores, en estos dos se observa un problema palpable que se repite en el anterior de MOG2, y es la cercanía y la saturación de la iluminación. Como se observa en la imagen de abajo, figura 59, solo hay dos murciélagos, pero ambos murciélagos están marcados por varias cajas delimitantes indicando que hay más de uno. El problema se ve más claramente en la imagen de la máscara umbral, donde se ve que el cuerpo del murciélago no está compuesto de un bloque conjunto, sino que tienen huecos negros indicando que no hay cambios, el lado derecho del murciélago está totalmente fragmentado en lugar de generar un bloque como hacía en videos anteriores, el cambio de intensidad en varios de los pixeles se detecta como insuficiente, de ahí que tenga tantos centroides y cajas.

Figura 59: GMG prueba 6

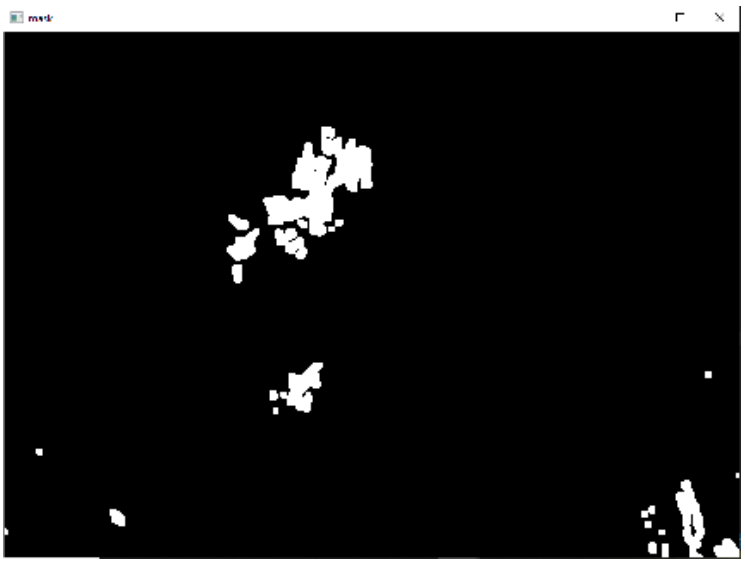


Figura 60: GMG prueba 6 mask

5.2.2 Análisis cuantitativo

En esta sección se llevará a cabo un análisis cuantitativo del funcionamiento de los sistemas mostrados en el apartado anterior. Se ha descartado aplicar estas pruebas a los métodos de coincidencia de formas, puesto que todos sus valores serían prácticamente 0. Este análisis tiene como objetivo mostrar la calidad de cada uno de los sistemas de background y de Haar-cascade.

Cada sistema está conformado por tres partes, cada una de las cuales tiene una tarea diferente, y por tanto la lógica diría que cada una requiere de medidas distintas para ser evaluado. El tracker es el mismo en todos los sistemas por lo que evaluarlo no tiene sentido ya que su funcionamiento está ligado al del detector.

Así pues se medirán los valores para los sistemas de detección y los resultados del conteo por separado.

Valores de conteo

Como ya se comentó cuando se explicó el diseño de la función de conteo, el sistema registra tres valores como cómputo global.

- ID: El número de identificadores distintos que se han asignado. Es un valor poco preciso a la hora de contar a los murciélagos, ya que cualquier ruido detectado se marca con un nuevo id. Pero es un valor indirecto que ilustra muy bien cuanto ruido ha cogido el detector cuanto más grande más ruido.
- Murciélagos de Salida: Es el número de murciélagos que se salen de los límites de la pantalla. Una vez más la mayor parte del ruido se genera en los márgenes del video, por lo que este valor también se ensucia mucho, y además no cuenta la totalidad de los murciélagos ya que la mayoría no sale de los márgenes. Este valor es más útil como otra marca de ruido.
- Murciélagos de entrada: Este es el valor que más importa, es el que registra a los murciélagos que salen de la entrada y entran a la misma, evitando contarlos más de una vez, siempre que estén marcados con el mismo id.

Todas las mediciones se han realizado varias veces por cada video, y se ha realizado una media con ellos, la mayoría de las veces se arroja el mismo valor, pero dado que la zona de salida se establece manualmente, a veces hay pequeñas variaciones en los datos.

Cada fila de la tabla representa una de las mediciones manos la última que es el número de murciélagos. Las columnas son los videos.

Recuento Haar-Cascade

	Video 1	Video 2	Video 3	Video 4
ID	48	1195	62	34
Murciélagos Salida	33	495	21	15
Murciélagos Entrada	8	276	21	8
Murciélagos Real	182	594	358	91

Figura 61: recuento Haar

Lo primero que hay que remarcar es que ninguno ha acertado el valor real, era esperable, ya se ha visto que los sistemas estaban lejos de ser perfectos.

Pero el caso de Haar es especialmente llamativo, por lo poco que se acercan los valores obtenidos a los valores reales, esto está causado por un solo motivo y es la inconstancia del detector. Como ya se mencionó en el análisis cualitativo, el método de Haar-cascade rara vez marca un mismo objeto durante más de tres o cuatro fotogramas seguidos, cuando la lógica dice que si el objeto se ha reconocido una vez, porque no una segunda.

Esto no es así por el sistema que emplea Haar para detectar los objetos, en el caso de imágenes tan imprecisas y poco distinguibles como las que se manejan en este dominio es muy fácil que uno de los estados etiquete la imagen como no objeto y que se rechace. Si la imagen es muy clara pasará varias veces todos los estados, pero eventualmente será rechazada.

Por eso hay tan pocos murciélagos obtenidos de entrada, los que se deberían contar cuando están cerca de la entrada a menudo no tienen historial de puntos anteriores por lo que no se considera que hayan tenido un desplazamiento según el método de conteo, y los bajos id son por el método de tracking que recicla los id puesto que desaparecen demasiados objetos entre fotogramas.

Es un resultado muy pobre, pero ilustra muy bien la falta de potencia de estas técnicas para reconocer objetos tan poco definidos y prácticamente indistinguibles.

Recuento MOG2

	Video 1	Video 2	Video 3	Video 4
ID	461	2159	1287	2865
Murciélagos Salida	72	1184	801	1205
Murciélagos Entrada	162	815	382	109
Murciélagos Real	182	594	358	91

Figura 62: recuento MOG2

Como ya anticipaban los análisis cualitativos los valores de salida y especialmente de id muestran la gran cantidad de ruido que hay en los videos especialmente en el video 4 que muestra una asignación de id enorme. Gran parte de este ruido se podría eliminar si se recortase el video en los márgenes o no se aplicase el detector en esas regiones. Pero ese valor es en realidad muy útil.

El primer video es el único que ha contado el número de murciélagos inferior al real. Esto es debido a la dimensión de la entrada de los murciélagos y la forma en la que estos salen. La entrada de la cueva es muy amplia y los murciélagos no siempre salen en línea recta, la mayoría de las veces dan vueltas y se cruzan unos con otros. Esto es de lleno la mayor debilidad del seguimiento por centroide, cuando se cruzan dos objetos hay bastante probabilidad de que intercambien su ID, de esta forma un murciélago que no había sido contado puede recibir el id de uno que si ha sido etiquetado como contado dando como resultado menos murciélagos de los que hay realmente.

Los otros tres videos muestran una detección de murciélagos mayor que el número real, lo hacen por causas similares, pero no iguales. Como ya se explicó en el análisis cualitativo, cuando los murciélagos están muy cerca de la cámara, tienden a volverse borrosos y algunas partes transparentan ya que se mueven a gran velocidad, estas transparencias se traducen en zonas que no cambian lo suficiente para el detector y por tanto el murciélago no se detecta como un ente unido sino que está fragmentado en varias partes, y algunos murciélagos se cuentan como dos o tres. Esto es más palpable mirando los datos de murciélagos de salida, en estos videos aproximadamente la mitad de los murciélagos salen de los límites de la pantalla, asumiendo que parte es ruido del entorno, sigue habiendo más del doble de conteos, es porque estos murciélagos salen de la pantalla divididos en varios centros.

En el video 2 ocurre algo similar. Pero aquí no hay fragmentación, los murciélagos están muy lejos y se engloban correctamente como uno, el problema es el ángulo de la iluminación. En este escenario la salida es una gran cueva con una pared casi blanca vista de lateral, la luz hace que los murciélagos emitan sombras más grandes incluso que ellos y que se ven mejor, tanto es así que a pesar del control de sombras de MOG2, las detecta igualmente, así a veces algunos murciélagos cuentan por dos.

En general los valores de conteo no están tan alejados de la realidad como los valores de id y salida dan a entender. Esto es principalmente gracias la función de conteo.

Recuento GMG

	Video 1	Video 2	Video 3	Video 4
ID	1233	4089	1971	3613
Murciélagos Salida	175	1493	1193	775
Murciélagos Entrada	156	987	371	162
Murciélagos Real	182	594	358	91

Figura 63: recuento GMG

Ya de entrada se ve algo que el análisis cualitativo ya anticipaba, los números de salida y especialmente de id muestran la cantidad de ruido que coge GMG, aunque gran parte es en los momentos iniciales, eso es verdad.

Por lo demás demuestra los mismos errores que sufre MOG2 per con el añadido de que capta más ruido. Si es curioso el caso del video 3 en el que a pesar de captar tanto ruido, ha sido más preciso a la hora de aislar a los murciélagos en la entrada y no los ha fragmentado tanto como su predecesor, aunque la diferencia es prácticamente ínfima.

Valores de rendimiento

Ya hemos visto cómo funcionan los detectores en conjunto con la función de conteo, pero para medir correctamente su eficacia se necesita recopilar otros datos diferentes.

Para ello se ejecutarán cada uno de los sistemas sobre cada uno de los videos obteniendo en cada una distintas medidas que se han considerado indicativas de funcionamiento, escogidas mirando el artículo de *"Comparative evaluation of background substraction algorithms in remote scene videos captured by MWIR sensors"* [17]. Donde se comparan varios algoritmos de extracción del fondo, pero cuyas medidas bien pueden adaptarse también para el caso de Haar.

Como se muestra en la figura 64, los valores que se van a medir son TP (true positives) verdaderos positivos, murciélagos que el detector marque como murciélagos. TN (true negatives) verdaderos negativos, aquellos que no sean murciélagos y que el sistema no marque como tales. FN (false negatives) falsos negativos, aquellos murciélagos que el sistema no marque como tales. FP (false positive) aquello que el sistema marque como murciélago y no lo sea.

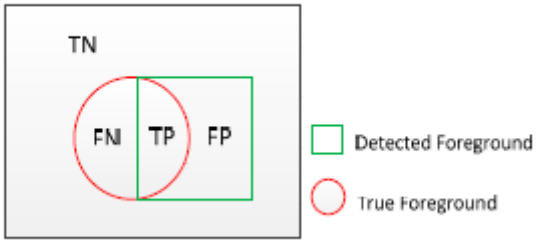


Figura 64: métricas de detección

Con estos datos además se podrán calcular las medidas de precisión, que indica la exactitud o calidad del sistema, el recall, que indica la completitud del sistema. Y la F-medida, un valor que indica la armonía entre la predicción y el recall, cuanto más alto el valor mejor es el sistema.

$$Precision = TP/(TP+FP)$$

$$Recall = TP/(TP+FN)$$

$$F-measure = 2 * Recall * Precision/(Recall + Precision)$$

Antes de mostrar los datos se van a explicar unas premisas que se han decidido a la hora de realizar las mediciones.

La primera es que se contarán como Verdaderos positivos las ocasiones en las que el detector marque una parte de un murciélago cuando este está fragmentado, es decir si marca el cuerpo y un ala, ambos se tomarán como verdaderos positivos.

En el caso de GMG no se han contado las detecciones que realiza antes de tener el modelo calibrado, es decir no se toman en cuenta los primeros frames.

Y tercero, en el caso de los verdaderos negativos, se incluyen todos los objetos que se mueven de un fotograma a otro y no son marcados por el detector, tales como plantas o partículas.

Por último, decir que estas medidas han sido tomadas de la única forma posible, a mano por una sola persona, por lo que el error humano es esperable en la toma de las mismas. También añadir que dada la enorme cantidad de tiempo que tomar estas medidas consume, únicamente se han tomado dos veces y se ha realizado una media entre ambas.

Medidas Haar-Cascade

	V1	V2	V3	V4
TP	126	564	421	89
TN	1958	4124	3412	1134
FP	559	1623	1263	264
FN	743	2106	1789	341
Precisión	0,1839	0,2578	0,25	0,2521
Recall	0,1449	0,2112	0,1904	0,2069
F-medida	0,1621	0,2322	0,2162	0,2273

Figura 65: medidas Haar

Sorprendentemente aquí el modelo de Haar-cascade sale mucho mejor parado que en la medición anterior, especialmente en los últimos tres videos, donde muestra un gran número de aciertos. No es tan sorprendente si se piensa que en cada frame el modelo señala dos o tres objetos, de los cuales al menos uno o dos suelen ser murciélagos, aunque en el siguiente no lo marque de nuevo, el número de verdaderos positivos es bastante elevado. Especialmente en los videos dos y cuatro donde prácticamente marca correctamente a todos los murciélagos mientras se mantienen dentro de la ventana.

Sin embargo el valor de recall llama la atención por lo bajo que es, se recuerda que este valor marca la completitud, cuanto del total detecta el sistema. Para este detector el valor es pequeño, dado que deja muchas cosas sin detectar.

Medidas MOG2

	V1	V2	V3	V4
TP	588	2738	1646	404
TN	1378	5194	3504	8470
FP	2118	10426	5712	1682
FN	190	788	468	28
Precisión	0,2172	0,2079	0,2237	0,1936
Recall	0,7557	0,7765	0,7786	0,9351
F-medida	0,3375	0,3281	0,3475	0,3208

Figura 66: medidas MOG2

La precisión del modelo ronda más o menos el 20%, es decir que de cada cinco cosas que detecta una es un murciélago, no parece muy bueno. El valor de Recall es en la mayoría de videos del 80% aproximadamente, eso quiere decir que detecta la mayoría de los murciélagos del video. El valor de la F-medida es mejor cuanto más alto es, un 30% no parece ser muy bueno.

Para ver si los valores son aceptables o no se tiene que juzgar en términos de algoritmos de Background substraction. En el mismo estudio al que se hizo referencia para obtener estas medidas, se muestran los valores de varios de los algoritmos de esta técnica. Entre

ellos se incluye MOG2 y sus valores son bastante parecidos a los que se han obtenido aquí. De hecho aquí se ha obtenido una mejor valoración

Medidas GMG

	V1	V2	V3	V4
TP	846	2514	1570	366
TN	1878	5384	4038	6312
FP	3260	10246	6316	1928
FN	226	604	396	28
Precisión	0,2060	0,1970	0,1990	0,1595
Recall	0,7891	0,8062	0,7985	0,9289
F-medida	0,3267	0,3166	0,3187	0,2723

Figura 67: medidas GMG

El modelo de GMG como podía esperarse ha obtenido una precisión menor que el MOG2 aunque su completitud es mayor. Este modelo también tiene unas medidas no muy lejanas de las del otro estudio, sin embargo este ha obtenido peores resultados en general.

Pero lo que interesa no es compararlos con los antiguos o entre ellos. De hecho lo ms interesante que arrojan estos datos es lo que ambos métodos tienen en común, y es que el video 4 se sale de las medidas de los otros tres en ambos métodos, destacando como en el que menos preciso han sido ambos pero mayor completitud han obtenido, y es que aunque el video contenga tanto ruido que da lugar a múltiples falsos positivos, hay muy pocos murciélagos en él y por lo tanto menor proporción de precisión. Con lo cual se puede deducir que si en los videos el número de murciélagos fuese mayor, la precisión y demás valores mejorarían notablemente.

En definitiva, el sistema de Haar se puede confirmar que es poco efectivo, seguramente debido a la falta de precisión del propio algoritmo para tratar con videos como el que se emplea en este proyecto.

Los algoritmos de background aunque pueda parecer que tampoco han cumplido del todo con las necesidades, si han actuado lo mejor que un algoritmo de background lo puede hacer, captan la práctica totalidad de los murciélagos aceptando bastante poco ruido en el proceso. Indudablemente cumplen la tarea lo mejor que esta clase de algoritmos pueden.

6. GESTIÓN DEL PROYECTO

6.1 Planificación

En esta sección de la memoria detallara la planificación que se ha seguido para realizar este proyecto.

En este tipo de trabajos a largo plazo es vital planificar con antelación y estimar los tiempos con el fin de que el proyecto se desarrolle satisfactoriamente. Es importante tener en cuenta todo lo que esto conlleva, hay tener presente los momentos de descanso, horas máximas de trabajo, y cualquier otro aspecto que pueda influenciar el periodo del proyecto.

La planificación que se estableció sobre este proyecto se realizó conforme a las diferentes tareas globales para realizar la misma. Cada una de las fases lleva asociado una estimación de tiempo acorde a su magnitud estimada:

1. Estudio del problema: se analiza el objetivo a realizar y se estudian los conocimientos que se requerirán para lograr cumplirlos. Se realiza una exhaustiva documentación para formarse en el problema y en las herramientas que se van a emplear para realizar el mismo. Dada la magnitud de los campos que tocan lo que se trabaja en el proyecto y la gran cantidad de nuevas investigaciones en los últimos años, se estimó un tiempo de 20 días más 7 adicionales en caso de imprevistos o una primera mala estimación.
2. Diseño de los algoritmos: Se deciden que algoritmos encajan con el problema a tratar y son viables con las herramientas que se manejan. Se diseñan las partes que compondrán cada uno de los sistemas y se ensamblan. Se estimó un tiempo de no menos de 30 días más otras 7 de margen por imprevistos o cambios en las decisiones.
3. Realización del documento y pruebas: Se documenta y ordena todo el trabajo que se ha realizado decidiendo que partes merecen más extensión. Se recopilan los datos del funcionamiento de cada uno de los sistemas mediante pruebas de rendimiento. Se documentan una vez terminadas. Se estiman 30 días de trabajo y 10 adicionales por imprevistos y correcciones.

El balance final se declara en $16 + 30 + 30 = 80$ días de trabajo, más $7 + 7 + 10 = 24$ días extra para cubrir imprevistos, errores, retrasos o necesarios cambios y rectificaciones. En total 104 días de trabajo.

El proyecto se empezó el Lunes 20 de Junio de 2019, estableciendo una primera fecha de finalización el día 5 de Septiembre de 2019. Aunque la límite fijada es el día 20 de Septiembre.

La Planificación como se mostrará a continuación se ha cumplido empleando 5 de los días extra.

A continuación se representará la planificación con cada una de las tres actividades desglosadas en varias sub actividades en un gráfico de Gantt, figura 68.

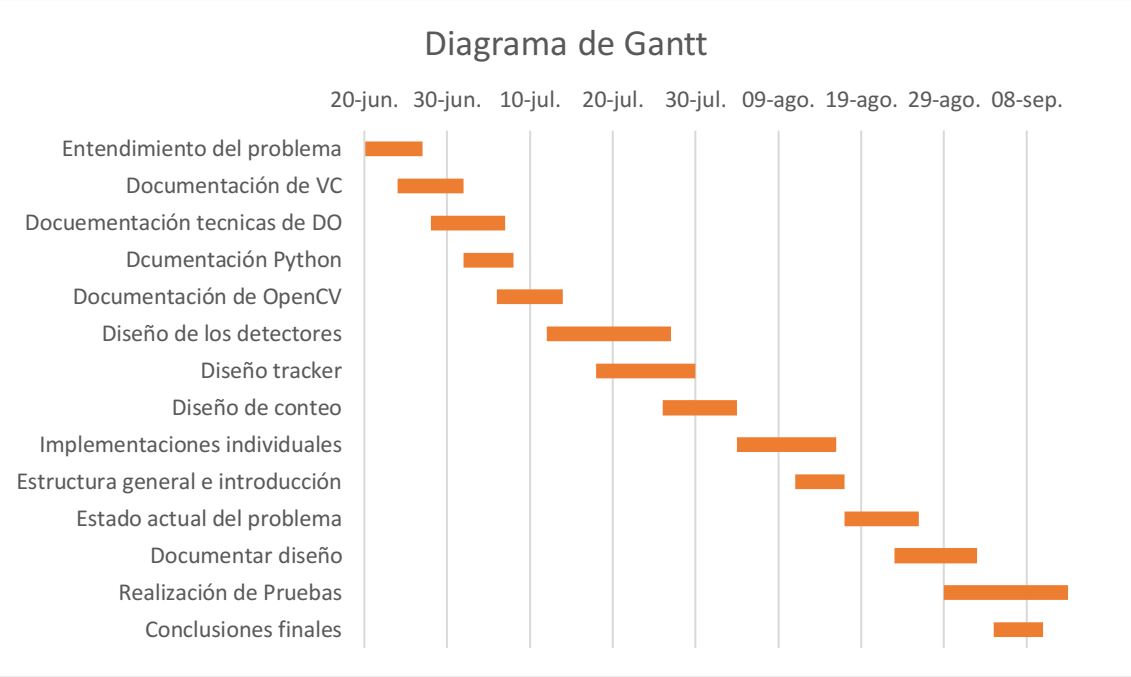


Figura 68: diagrama de Gantt

La gráfica corresponde con la tabla:

NOMBRE ACTIVIDAD	FECHA INICIO	DIAS	FECHA FIN
Entendimiento del problema	20-jun	7	27-jun
Documentación de VC	24-jun	8	02-jul
Documentación técnicas de DO	28-jun	9	07-jul
Documentación Python	02-jul	6	08-jul
Documentación de OpenCV	06-jul	8	14-jul
Diseño de los detectores	12-jul	15	27-jul
Diseño tracker	18-jul	12	30-jul
Diseño de conteo	26-jul	9	04-ago
Implementaciones individuales	04-ago	12	16-ago
Estructura general e introducción	11-ago	6	17-ago
Estado actual del problema	17-ago	9	26-ago
Documentar diseño	23-ago	10	02-sep
Realización de Pruebas	29-ago	15	13-sep
Conclusiones finales	04-sep	6	10-sep

Figura 69: tabla de actividades

6.2 Costes

En esta sección se realizará el análisis del balance monetario del proyecto en término de costes.

El presupuesto del proyecto está dividido en dos partes:

- Coste humano: el gasto generado por el personal humano contratado para este proyecto.
- Coste material: el gasto generado por los materiales, equipo empleados y licencias de software.

COSTE HUMANO

Nombre	Coste/hora	Coste/día	Duración trabajo	Coste final
Jesus García Cuchillero	10 €/h	80 €/h	85 días	6800 €

Figura 70: coste humano

COSTE MATERIAL

Descripción	Coste material/licencia	Duración	Uso en el proyecto	Amortización	Coste imputable
Ordenador: HP	420 €		100%	48 meses	26,05 €
Licencia Windows 10	135 €		100%	24 meses	4,57 €
Licencia Microsoft Office	539 €		30%	48 meses	9,36 €
				TOTAL	39,98 €

Figura 71: coste material

COMPUTO TOTAL

CONCEPTO	COSTE
Coste Humano	6.800 €
Coste material	39,98 €
Sin impuestos	6.839,98 €
IVA	21%
Coste proyecto	8.276,38 €

Figura 72: coste total

7. ENTORNO SOCIOECONÓMICO

Como ya se adelantó en la introducción, en el apartado del marco regulador, este proyecto está más enfocado a la investigación. Es más un paso previo para trabajos futuros, se espera que se use como apoyo para elaborar sistemas más avanzados y sirva de base para quien realice ese proyecto.

En el supuesto de que los sistemas obtenidos en este proyecto se empleasen, se podría considerar un impacto ambiental. Como ya se ha comentado, este proyecto nació a raíz de la necesidad de llevar un control de las poblaciones de murciélagos de la península, pero el proceso para realizar los conteos es extremadamente tedioso y consume mucho tiempo. El empleo de este tipo de sistemas podría ahorrar tiempo a los trabajadores que realizan esta tarea. A sabiendas de que el método es automático podrían tomar más videos y de mayor duración lo que permitiría llevar un control más preciso de las poblaciones algo beneficioso para el medio ambiente y el ecosistema, y además podrían aprovechar el tiempo que gastan en esta tarea para realizar otras.

Pero en un principio este proyecto en concreto no es para el empleo directo, con lo que no hay planes de explotación para el mismo, ni impactos económicos importantes. Es para uso interno.

8. CONCLUSIONES Y TRABAJOS FUTUROS

En este proyecto buscaba alcanzar dos metas, por un lado se quería construir uno o varios sistemas que fuesen capaces de realizar un conteo de murciélagos en vuelos nocturnos mediante el uso de técnicas de la etapa tradicional de la visión artificial. El segundo era juzgar a partir de los sistemas diseñados, si la tarea se resolvía de forma satisfactoria con este tipo de técnicas o si era una tarea que requería de técnicas de la segunda etapa.

Los resultados obtenidos en las pruebas son poco satisfactorios en lo que al rendimiento se refiere. Los resultados de detección que han mostrado son muy bajos para poder considerarlos como una detección satisfactoria, al menos considerando la precisión que exhiben otros sistemas existentes más elaborados. Sin embargo son altamente esperanzadores, resuelven la duda de si merece la pena intentar implementar los sistemas más modernos y potentes de Deep learning, ya que si con estas técnicas se ha obtenido estos resultados, indudablemente es posible incrementarlos.

El mayor obstáculo que se ha encontrado en este proyecto es sin lugar a dudas la vastedad del campo que abarca y el desconocimiento total que se tenía de la materia. Se eligió este proyecto porque no se había trabajado nunca con nada relacionado con la visión artificial y este proyecto prometía ser una forma muy buena de empezar y adquirir conocimientos acerca del mismo y probarlo. Lo que a priori podría parecer la mayor ventaja de esta materia es sin lugar a dudas su mayor dificultad. Es un campo de investigación que actualmente está en su apogeo y que por muchos años ha sido objeto de numerosos estudios, artículos y publicaciones, en definitiva hay material más que de sobra para documentarse acerca del tema. Sin embargo es tanta la cantidad de investigaciones y ramas exploradas, que es muy difícil, hacerse una idea general de la materia, sencillamente hay demasiadas opiniones, interpretaciones y acercamientos a los problemas de la visión computacional. Cada una de las tareas que conforman este campo de la tecnología pueden ser resueltas de infinidad de maneras completamente diferentes para obtener los mismos resultados, son tantos los enfoques que cuando se empezó el proyecto fue un poco abrumador. Era como ver la punta del iceberg pensando que era muy grande y dase cuenta de que debajo es aún mayor. Sin duda la parte más difícil fue entender la línea general, escoger el punto de partida y decidir que opciones explorar para no perderse entre tanta información.

Una vez pasado ese punto el proyecto se volvió algo más manejable y concreto, y las cosas fueron surgiendo mejor.

No hay ningún tipo de arrepentimiento por haber escogido este trabajo, la experiencia ha sido enormemente enriquecedora, se han trabajado con conceptos que no se habían manejado nunca y se han logrado manejar con éxito, se han adquirido una gran cantidad de competencias que serán de gran utilidad en el futuro personal. Pero eso no significa que no haya un pequeño sentimiento de frustración.

Ya se ha dicho que el resultado deja mucho que desear, ahora que se han comprobado cómo funcionan estas aproximaciones, se desearía probar con más métodos e intentar mejorar los resultados obtenidos, quizás incluso combinaciones entre los acercamientos que no se plantearon de buen principio. En general se considera que con los

conocimientos que ahora se poseen se podría lograr alcanzar mejores resultados que los obtenidos, hay margen para mejora.

Pero no se desea mostrar una visión pesimista acerca del proyecto, ya que este marca un precedente excelente para trabajos futuros que se han considerado realizar quizás incluso este mismo año o el siguiente.

Se ha planteado realizar un proyecto similar pero centrándose en técnicas más modernas basadas en CNN y ver qué resultados pueden obtenerse con los mismos. El trabajo realizado en este proyecto bien puede servir para pre procesar las imágenes de las que se servirán estos sistemas para entrenar sus modelos, y como base para alguien que también acabe de empezar.

9. BIBLIOGRAFÍA

[1] https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3

[2] https://www.hisour.com/es/computer-vision-42799/

[3] "Object Detection in 20 Years: A Survey" Zhengxia Zou, Zhenwei Shi,Member, Yuhong Guo, and Jieping Ye,Senior Member.

https://arxiv.org/pdf/1905.05055.pdf

[4] "Systematic Survey on Object Tracking Methods in Video" J.Joshan Athanesious, P.Suresh

https://pdfs.semanticscholar.org/2e68/86acf77baa00b59e3b4a8361abac61e344bf.pdf

[5] https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

[6] https://www.quora.com/What-are-some-interesting-applications-of-object-detection

[7] "Automatic detection, tracking and counting of birds in marine video content"

https://ieeexplore.ieee.org/document/7821031

[8] "A combined corner and edge detector" Chris Harris & Mike Stephens

http://www.bmva.org/bmvc/1988/avc-88-023.pdf

[9] "Good features to track" Jianbo Shi, Carlo Tomasi

http://www.ai.mit.edu/courses/6.891/handouts/shi94good.pdf

[10] "Distinctive ImageFeaturesfromScale-InvariantKeypoints" David G. Lowe

https://people.eecs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf

[11] "Machinelearningfor high-speed cornerdetection" Edward Rosten and Tom Drummond

https://www.edwardrosten.com/work/rosten_2006_machine.pdf

[12] "BRIEF: Binary Robust IndependentElementary Features" Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua

https://www.cs.ubc.ca/~lowe/525/papers/calonder_eccv10.pdf

[13] "ORB: an efficient alternative to SIFT or SURF" Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski

http://www.willowgarage.com/sites/default/files/orb_final.pdf

[14] "RapidObjectDetectionusinga BoostedCascadeofSimpleFeatures" Paul Viola, Michael Jones

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[15] “An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection” P. KaewTraKulPong and R. Bowden

http://personal.ee.surrey.ac.uk/Personal/R.Bowden/publications/avbs01/avbs01.pdf

[16] “Visual Tracking of Human Visitors underVariable-Lighting Conditions for a ResponsiveAudio Art Installation” Andrew B. Godbehere, Akihiro Matsukawa, Ken Goldberg

https://goldberg.berkeley.edu/pubs/acc-2012-visual-tracking-final.pdf

[17] “Comparative Evaluation of Background Subtraction Algorithms in Remote Scene Videos Captured by MWIR Sensors” Guangle Yao, Tao Lei, Jiandan Zhong, Ping Jiang and Wenwu Jia
https://www.researchgate.net/publication/319280159_Comparative_Evaluation_of_Background_Subtraction_Algorithms_in_Remote_Scene_Videos_Captured_by_MWIR_Sensors

10. ANEXO

The main objective of this proyect is to analyze different tecniques and algorithms for tracking and detecting objects in images and videos belonging to the clasical aproach of Computer visión, in order to understand how they work and be able to adapt and apply them to night recordings of complex flights. And thus determine how effective these techniques are for this particular domain.

For the study and test, implementations based on the Python language will be used, together with the OpenCV open source library of artificial vision. With the implementation, different tests will be carried out to assess whether the result is satisfactory and draw the relevant conclusions for this work.

1. INTRODUCTION

1.1 Work motivation

The idea for this work arises in a conversation between the tutor of this TFG and his roommates with some acquaintances who work in the pest control of the peninsula.

These people among other jobs are in charge of keeping track of the number of bats on the peninsula. For this they travel to different places where there are colonies of bats and during the night they place a night camera focusing the entrance of the colony, and with some sound systems they make them leave the cave little by little. Bats that leave the cave and enter, are recorded in the chamber, and workers with a manual counter pause the video sequence by sequence, counting the bats and following them on the screen trying not to count them several times. This way of doing it as you can imagine is very slow and tedious.

So with the enormous boom that monitoring and detection techniques are having in recent years, why not try to create an application that could detect bats and count them in the same way that these people do, but automatically.

This was the initial idea on which this TFG was raised.

1.2 Work objectives

The reasons given in the previous section make clear the "why", that is, the need that has led to the realization of this work, but it is also necessary to know the "what", which is exactly what is needed and how it can be satisfied.

The answer to that question is where the objectives that will set the guidelines for carrying out the project are extracted.

In recent years the detection applications have had immense growth and development. They have evolved greatly in both accuracy, speed and efficiency. However, these new systems are heavy to create and very difficult to calibrate. Not only do they require powerful electronic equipment to function, but if they want to operate properly, huge amounts of data are needed to train them. In cases of everyday objects, such as people, cars, dogs ... there are huge databases available to everyone, and there are even models already trained in these same banks. But the problem that arises in this document is not common, the videos that are treated have unique characteristics and therefore a model tailored to the problem should be created.

That is why it was first set out to see how older techniques work, but much lighter and more varied. If its operation were promising, these same systems could be used to collect data in future projects with more modern systems. In this way these works would be raised on a solid basis and would have an easy way to extract information from those data that they need in order to properly function. Thus, the objectives of this project were marked as:

1. Investigate the different techniques and methods of video analysis and detection of existing objects in the classical stage. When talking about research, there is no reference only to the understanding of its operation and implementation, it is imperative to be able to determine theoretically if the algorithm or method is viable or not to try to solve the problem that arises.

   To achieve this, the different operations and functions applied by each of them must be analyzed and adapted so that their operation adapts to the characteristics of the problem in question.

2. Get to create at least one or several systems that use these techniques for tracking and detecting objects on video and check if you are able to identify and track the flight of bats in night camera recordings (gray tones) with good performance, both in speed as in percentage of success.

   In this objective it is important to distinguish the two parts that must compose the program so that it can meet this objective.

   - The detection or identification is the part of the program that should be able to, given an instance of the video, determine how many of the desired objects are in that instance and also in what position they are.
   - The tracking or tracking, is the part that given the positions of each object by the detection algorithm must be able to mark each one individually and follow it in the successive instances of the video, being robust to losses, on the other hand, by the Detection algorithm or by the output of video objects.

This could almost be established as two different objectives, but since they do not meet any real objective alone, it is the assembly of both parties that will generate the optimal solution.

3. Get the program to keep track of the number of objects (bats) that go through the videos. This is the final objective and the most important, the resulting code must

keep the individual account of the objects as accurately as possible, that is, be able to distinguish between the detected objects enough to not count them more than once, while these remain on video, increasing the overall count each time a new individual appears. Again, this must be robust against losses.

2. PROBLEM ANALYSIS

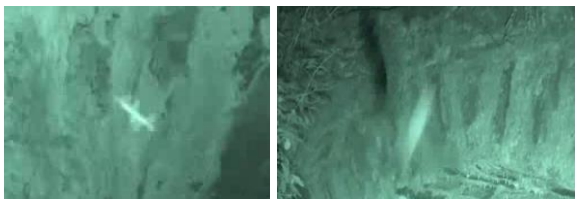
The problem that we are trying to solve in this document is to investigate different algorithms of the traditional stage of artificial vision and implement one or several systems that using these techniques are able to count the total number of bats flying in the recordings provided at night, in order to facilitate this counting task to those responsible for pest control and determine if it is feasible to use more modern algorithms in future projects.

To achieve this goal, the problem faced and the tasks required to solve it must be analyzed.

The best point to start is to analyze the domain for which these systems will have to be designed. There are a total of 4 videos, which show three different locations of colonies of these animals. Two videos show the same location but differ in the flow of bats in one and the other.

Although all three environments share the type of recording and format, the camera pointed to the entrance and exit of the colony. They differ in several aspects that must be taken into account, the most striking are the shape and size of the entrance and the amount of objects around each one that can potentially hinder the task, objects that must be taken into account when designing the solutions. But they are not the only ones, factors such as the difference in lighting between them or the distance to it are also important and should be taken into account in the design phase.

It is also necessary to analyze the objects of interest in them. This are different cropped catches of bats that will have to be dealt with in this document:



In these images it is anticipated that the task will be complicated, in most cases the shape of the bat is barely hinted, in others they are mere brighter spots or blots. The fact that the recordings are nocturnal complicates it even more since you cannot count on colors to help distinguish them in any way. And the recordings do not have good resolution which makes it difficult to distinguish by specific features.

Having seen and examined the domain with which this project will be treated, the key question that needs to be answered is. How can it be done to count bats?
The answer to the question is much more intuitive than it seems. As a person would. If a person wanted to count the bats in the videos, the first thing he would do would be to look for them on the scene, see where they are and how many are there. To make sure that he doesn't count them twice, the person would have to follow them individually throughout the video until they were out of sight, once out of the video they can no longer know anything about the bats.

This is exactly what the system that is built to deal with this problem has to do, it should be able to see the bats in the video and follow their movements until it lose sight of them when it can no longer know more about them. These are the three parts that are needed to satisfy the problem, find the bats, follow them through the video and count them without repeating them.
After analyzing the problem, only the how is missing.

3. SOLUTION DESIGN

As already mentioned in the previous sections for this project, four possible solutions for the problem that arises will be designed.

The choice of this number of systems is based on the possible approaches that can be made towards it and the limitations of the tools used. When objects are detected in videos, you can focus it in two ways. You can try to locate and recognize different parts of the object or the entire object itself and be able to distinguish it from the rest of the element around it. Or you can try to find another feature that makes it unique in addition to its shape, such as color or movement.
The idea of designing and implementing several is to be able to test the different approaches to see how each one works. It is not the objective to make a comparison between them, but to see how viable each of the solutions is.
To solve the problem that has arisen the four systems must be composed of three distinct parts, namely: the detector, the tracker or tracking system and the counting function.

3.1 Detection systems

These functions are responsible for finding the object in the image and indicating its position. To adapt the characteristics of the problem described, they must be light enough systems to run on each of the frames of the video.

These are the methods that make the distinction between approaches.

Feature matching ORB:

ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First it use FAST to find keypoints, then apply Harris corner measure to find top N points among them. It also use pyramid to produce multiscale-features. But one problem is that, FAST doesn't compute the orientation. So what about rotation invariance? Authors came up with following modification.

It computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to centroid gives the orientation. To improve the rotation invariance, moments are computed with x and y which should be in a circular region of radius r, where r is the size of the patch.

Haar-Cascade

The Haar cascade algorithm is an object detection algorithm based on machine learning. It was proposed in 2001 as already mentioned, by Paul Viola and Michael Jones.

This approach is based on a cascading function that trains from a multitude of positive and negative images.

The algorithm has 4 states:

1. Selection of Haar traits
2. Creation of integral images
3. Adaboost training
4. Cascading classifiers.

This algorithm works with adjacent rectangular regions at a specific location in the detection window, adds the pixel intensities in each region and calculates the difference between those sums.

Background substraction : MOG2 and GMG

Background substraction is one of the most effective methods to identify objects, not because of how they are but because of their characteristic of movement in a video sequence and is in fact a fundamental step in many artificial vision systems and safety or traffic systems. Basically these subtraction algorithms need a stable background, which is often very complicated in real-life applications. In this method the video sequences are divided into frames and each frame is used as a reference to create a background model. The pixels of the current frame, which differ from the background model, are considered

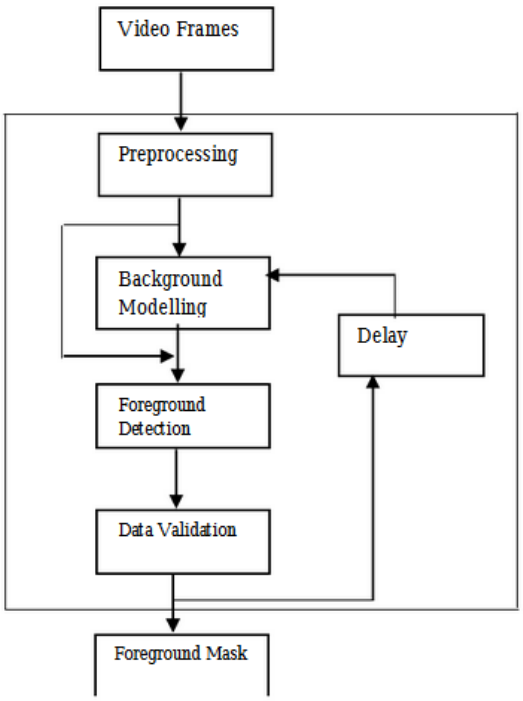
moving objects. For use in locating and tracking objects, more processing is needed on these objects called foreground. Since background extraction is usually the first step in any artificial vision application, it is crucial that the pixels extracted from the foreground correspond correctly with the moving object of interest.

The background substraction technique is a very simple algorithm but very sensitive to external changes in the environment, and is greatly affected by interference. This method can give the almost perfect information of the object that is sought if the whole fund is owned.

There are many challenges when developing a good background extraction algorithm, the algorithm must be robust to changes in lighting, should be able to avoid detecting parts of the background that are not stationary, such as leaves, rain, shadows, etc ... and the internal fund model should change rapidly if there is any change in the fund.

Base algorithm:

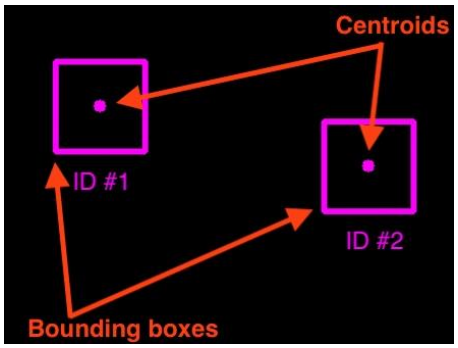
Many background extraction algorithms have been proposed in recent years, but identifying moving objects in a complex background remains a challenge. Most algorithms follow the same basic model consisting of four main steps; preprocessing ,background modeling, foreground detection and data validation.



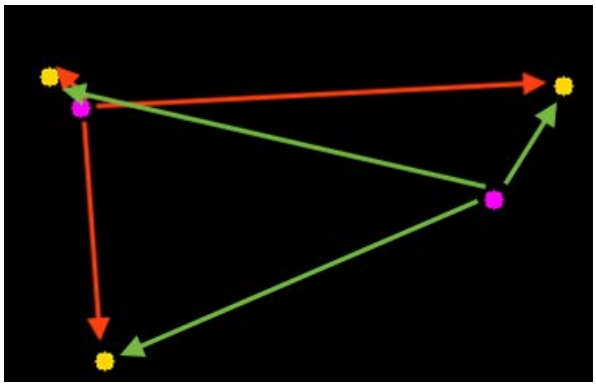
3.2 Tracking system

Assignment of centroid-based ID: It is an approximation that bases its operation on the distances between the centers, or centroids of an existing object and the centers of new objects that appear.
These types of algorithms are multi-process and their operation can be summarized in 3 steps.

1. The algorithm requires as input a bounding box that marks the position of the object to be followed (in this case these coordinates are given by the detection system). The method calculates the coordinate center of the box and assigns it an id identifier.

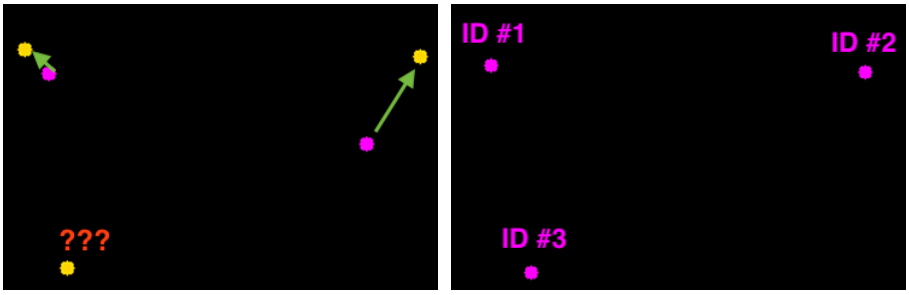


2. For each new frame of the video, the centroids will be searched again as in the previous step, however, instead of assigning new identifiers to each of the detections, it will try to associate the new centroid with the old one, the one of the previous frame. For this, the distance (Euclidean, Gaussian, etc ...) between the old and the new is calculated to try to associate them, with the assumption that the smaller the distance, the more likely they are to be the same object.



3. But new centers will always appear, or it will be impossible to relate two instances. When it is the case of not being able to relate a new centroid with an old one, it is registered as a new one.

Similarly, if a centroid disappears, either because the detector has lost it or left the screen, it is removed from the registry.



3.3 Count system

This function is the one that has to be in charge of keeping the bat account on screen as accurately as possible. Its design is based on the way a person would count the bats in the video.

Each video has a different distribution, but they all have two things in common, a point of entry and exit of bats to the colony and the limits of the video. Although the function has to work in all videos, it should be designed so that it takes advantage of the common characteristics of these.

When bats are counted by hand (with the video evidently slowed down) a person has easy to identify what each of the bats are, a person is the ideal detector and tracker, so the only thing the person who counts them is not aware of is what is outside the limits of the video, the interior of the colony and beyond the edges, during the rest of the time the state of the bat is known. In short, for an effective count the bat must be counted either when it appears on the screen or when it disappears from the screen by one of these boundary zones.

Summarizing it as it is in the implementation, the counting area is painted, if the position of a centroid in the current frame is within that area, the previous position is checked in its register (if it is not new to itself) continue as if nothing) if this is outside the boundary region, the position difference between the two for the x-axis and the y is looked at separately, if the absolute value of this difference is greater than 20 pixels on the x-axis, the Sign of the difference result, negative is labeled as rightward shift, positive to the left. The same for the y-axis, negative down, positive up.

The reason for these labels is that by default the objects are counted where they enter, but depending on the video, it can be desired that they only be counted if for example they come from the right.

4. RESULTS

Description of the experimental tests

In this section, several experimental tests will be carried out in order to verify the effectiveness and accuracy of the different systems implemented. The tests will be performed on the videos mentioned above.

The tests will first consist of a visual check of its operation, choosing each video as input and examining the points that the system captures by checking if it minimally fulfills the task of detecting and following the desired objects.

In addition, a comparative quantitative study will be carried out with the objective of measuring the performance of the different systems with each other. With this we want to obtain which system of the implemented ones performs the task more effectively and illustrate which one is the most suitable for the task.

Analysis

Qualitative analysis

In this section the qualitative analysis of the data will be carried out, which, as already mentioned, will consist of analyzing all the videos with each of the systems implemented and checking if the system detects and follows the objects of interest, if it is often wrong, as handles distant objects, occlusion, collision and other similar aspects.

Quantitative analysis

In this section a quantitative analysis of the operation of the systems shown in the previous section will be carried out. This analysis aims to show the quality of each of the background and Haar-cascade systems.
Each system is made up of three parts, each of which has a different task, and therefore the logic would say that each requires different measures to be evaluated. The tracker is the same in all systems so evaluating it makes no sense since its operation is linked to that of the detector.

Thus, the values for the detection systems and the counting results will be measured separately.

ORB centroid tracking: As already mentioned in previous sections, this approach has proved to be very poor for the videos discussed in this document. This detection system has problems, the main one is how little flexible your system is.

This causes the vast majority of bats not to be catch if they are not in similar positions to their reference model, especially between videos with differences in lighting.

Haar-Cascade: The second approach to object recognition and subsequent location, and the only one that uses learning techniques.
The truth is that although it is more powerful than the system of coincidence of forms and is much more adaptable to variations, it suffers from very similar problems. To train this system in the detection of bats, catches of all kinds were used, both from distant and nearby bats, sharper and less, in shadows, in illuminated areas, etc.
These tests already establish that the approach of recognition through cascade techniques lacks learning power to be able to discern that not all luminous points are bats, and their method of classification based on intensities suffers with bats in more illuminated areas.

Background MOG2: This detector has shown very good visual results. As explained in previous sections, a relatively high detection threshold was established for this algorithm, with the intention of eliminating small noises but allowing it to detect the changes of the farthest bats in the plane. It was also established that the shadows were distinguished, which avoids the noise of these in videos like 1 and also allows better capture blurred parts of a bat such as moving wings.

GMG Background: This combination has not shown the best results, although as seen in the images the system captures bats very well, it also does the same with the branches and particles that pass through the screen, it definitely captures too much noise. This is easily justifiable since, as mentioned in past sections, GMG needs to let several frames pass until it starts making good detections, since it uses those first frames to create a model with which to analyze the following ones. This generates two problems mainly, the first one that in the first seconds of the video playback the system captures practically everything as a movement, giving rise to many false and wrong measurements that alter the measurements. The second problem depends more on the video in question, take the video first, the V1, already in the first instances of video there are many bats moving around the screen, and the movement of the plants, which although it is slower than that of bats, in the time it takes GMG to calibrate, the movement of the plants is much more noticeable, this causes GMG not to have a sequence of “clean” frames, so to speak, of noise to generate a model faithful to that of the fund.

In short, Haar's system can be confirmed to be ineffective, probably due to the lack of precision of the algorithm itself to deal with videos like the one used in this project.
Background algorithms although it may seem that they have not fully met the needs, if they have acted the best that a background algorithm can do, they capture almost all bats accepting quite little noise in the process. They undoubtedly fulfill the task the best that this kind of algorithms can.

5. CONCLUSIONS AND FUTURE WORK

In this project I sought to achieve two goals, on the one hand we wanted to build one or several systems that were capable of counting bats on night flights by using techniques of the traditional stage of artificial vision. The second was to judge from the systems designed, if the task was solved satisfactorily with this type of techniques or if it was a task that required techniques of the second stage.

The results obtained in the tests are unsatisfactory in terms of performance. The detection results they have shown are very low to be considered as a satisfactory detection, at least considering the accuracy exhibited by other existing systems. However, they are highly encouraging, they resolve the question of whether it is worth trying to implement the most modern and powerful Deep learning systems, since if these techniques have obtained these results, it is undoubtedly possible to increase them.

The biggest obstacle that has been found in this project is undoubtedly the vastness of the field it covers and the total lack of knowledge of the subject. This project was chosen because it had never worked with anything related to artificial vision and this project promised to be a very good way to start and acquire knowledge about it and try it. What a priori might seem to be the greatest advantage of this matter is undoubtedly its greatest difficulty. It is a field of research that is currently in full swing and that for many years has been the subject of numerous studies, articles and publications, in short there is more than enough material to document the subject. However, there is so much research and branches explored that it is very difficult to get a general idea of the subject, there are simply too many opinions, interpretations and approaches to the problems of computational vision. Each of the tasks that make up this field of technology can be solved infinitely in completely different ways to obtain the same results, there are so many approaches that when the project started it was a bit overwhelming. It was like seeing the tip of the iceberg thinking it was very large and realize that underneath it is even bigger. Without a doubt, the most difficult part was to understand the general line, choose the starting point and decide what options to explore so as not to get lost among so much information.

Once that point passed, the project became somewhat more manageable and concrete, and things were emerging better.

There is no regret for having chosen this work, the experience has been enormously enriching, i have worked with concepts that had never been handled and i have managed to manage successfully, i have acquired a lot of skills that will be great usefulness in the personal future. But that does not mean that there is no small feeling of frustration.

It has already been said that the result leaves much to be desired, now that these approaches have been proven, one would like to try more methods and try to improve the results obtained, perhaps even combinations between the approaches that were not raised from the beginning. In general, it is considered that with the knowledge that is now possessed, it would be possible to achieve better results than those obtained, there is room for improvement.

But i do not want to show a pessimistic view about the project, as this marks an excellent precedent for future work that has been considered to be carried out perhaps even this year or the next.

It has been proposed to carry out a similar project but focusing on more modern techniques based on CNN and see what results can be obtained with them. The work done on this project may well serve to preprocess the images that these systems will use to train their models, and as a basis for someone who has also just started.